

```
SSSSSSSSSSSSSS 000000000 000 RRRRRRRRRRRR RRR TTTTTTTTTTTTTTT 333333333 222222222
SSSSSSSSSSSSSS 000000000 000 RRRRRRRRRRRR RRR TTTTTTTTTTTTTTT 333333333 222222222
SSSSSSSSSSSSSS 000000000 000 RRRRRRRRRRRR RRR TTTTTTTTTTTTTTT 333333333 222222222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSSSSSSSSS 000 000 RRRRRRRRRRRR RRR TTT 333 333 222 222
SSSSSSSSSS 000 000 RRRRRRRRRRRR RRR TTT 333 333 222 222
SSSSSSSSSS 000 000 RRRRRRRRRRRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSS 000 000 RRR RRR TTT 333 333 222 222
SSSSSSSSSSSS 000000000 000 RRR RRR TTT 333333333 222222222222222
SSSSSSSSSSSS 000000000 000 RRR RRR TTT 333333333 222222222222222
SSSSSSSSSSSS 000000000 000 RRR RRR TTT 333333333 222222222222222
```

SR
VO

```

LL                      IIIII
LL                      IIIII
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LL                      II
LLLLLLLLLLLL           IIIIII
LLLLLLLLLLLL           IIIIII

SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSSS
SSSSSSSSS

```



```
1 0001 0 MODULE SRTRN( MAIN = SRTRN,  
2 0002 0 IDENT = 'V04-000' ! File: SRTRN.B32 Edit: PDG3006  
3 0003 0 %IF %BLISS(BLISS32) %THEN  
4 0004 0 ADDRESSING_MODE(EXTERNAL=GENERAL)  
5 0005 0 %FI  
6 0006 0 ) =  
7 0007 1 BEGIN  
8 0008 1  
9 0009 1 *****  
10 0010 1 *  
11 0011 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
12 0012 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
13 0013 1 * ALL RIGHTS RESERVED.  
14 0014 1 *  
15 0015 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
16 0016 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
17 0017 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
18 0018 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
19 0019 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
20 0020 1 * TRANSFERRED.  
21 0021 1 *  
22 0022 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
23 0023 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
24 0024 1 * CORPORATION.  
25 0025 1 *  
26 0026 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
27 0027 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
28 0028 1 *  
29 0029 1 *  
30 0030 1 *****  
31 0031 1  
32 0032 1  
33 0033 1 ++  
34 0034 1  
35 0035 1 FACILITY: VAX-11 SORT/MERGE & PDP-11 SORT/MERGE  
36 0036 1  
37 0037 1 ABSTRACT:  
38 0038 1  
39 0039 1 This module translates specification files,  
40 0040 1 from:  
41 0041 1 SORT-11 V2.0 or VAX-11 SORT/MERGE V3.0 format  
42 0042 1 to:  
43 0043 1 PDP-11 SORT/MERGE V3.0 or VAX-11 SORT/MERGE V3B format.  
44 0044 1  
45 0045 1 ENVIRONMENT: VAX/VMS user mode  
46 0046 1  
47 0047 1 AUTHORS:  
48 0048 1 Peter D Gilbert, Victor L Bennison, CREATION DATE: 21-Jan-1982  
49 0049 1  
50 0050 1 MODIFIED BY:  
51 0051 1  
52 0052 1 T03-001 Original  
53 0053 1 T03-002 Distinguish between F and D floating. PDG 3-Mar-1983  
54 0054 1 T03-003 Major fixes, including variable-format support, and addition  
55 0055 1 of /S11 and /S32 qualifiers. PDG 3-Mar-1983  
56 0056 1 T03-004 Remove GLOBAL declarations. PDG 11-May-1983  
57 0057 1 T03-005 Changes so this can be used on PDP-11s. PDG 28-Sep-1983
```


T03-006 Added forced fields to the key list so they are now output in the correct order relative to other keys. Changed to GCML for Bliss-11. Added DTYPE AZ and DTYPE_DIBOL. Interpretation of 'Z' data type depend on 7S11/S32 qualifier. Check output LRL length. PDG 26-Oct-1983


```

65      0064 1 LITERAL
66      0065 1
67      0066 1 LITERAL K_NATIVE = 1; ! True to use native routines for VAXen
68      0067 1
69      0068 1 K_NEW = 1;
70      0069 1 LIBRARY 'SRC$:SFKEYWRD'; ! Keywords in specification file
71      0070 1 %IF %BLISS(BLISS32) %THEN
72      0071 1 LIBRARY 'SYSS$LIBRARY:STARLET'; ! Defines system services,etc.
73      0072 1 LIBRARY 'SYSS$LIBRARY:XPORT';
74      0073 1 REQUIRE 'SRC$:SRTTRNMSG'; ! Error message definitions
75      U 0171 1 %ELSE
76      UU 0172 1 LIBRARY 'SYSS$LIBRARY:RMSINT'; ! Defines system services,etc.
77      UU 0173 1 LIBRARY 'SYSS$LIBRARY:XPORT';
78      UU 0174 1 REQUIRE 'SRC$:SRTTRNMSC'; ! Miscellaneous
79      UU 0175 1 REQUIRE 'SRC$:SRTTRNMSG'; ! Error message definitions
80      U 0176 1 LIBRARY 'SYSS$LIBRARY:RSX11M';
81      0177 1 %FI
82      0178 1
83      0179 1 LINKAGE
84      0180 1 CALL = CALL;
85      0181 1 FORWARD ROUTINE
86      0182 1 SRTTRN, ! Main entry
87      0183 1 DEF_FIELDS: NOVALUE,
88      0184 1 DEF_SUBFIELDS: NOVALUE,
89      0185 1 PUT_DEFN: NOVALUE,
90      0186 1 PUT_NAME: NOVALUE,
91      0187 1 PUT_FFLD: NOVALUE,
92      0188 1 REPORT_IO_ERROR: CALL, ! Report on I/O errors
93      0189 1 INIT_IO: NOVALUE, ! Parses command line, opens and connects files
94      0190 1 CVT_DTB, ! Converts ascii to binary
95      0191 1 CVT_BTA, ! Converts binary to ascii
96      0192 1 NEW_FMT,
97      0193 1 DO_FIELD: NOVALUE, ! Get all field definitions if no record types
98      0194 1 DO_HEADER: NOVALUE, ! Get header information
99      0195 1 DO_ALTSEQ: NOVALUE, ! Process altseq line
100     0196 1 GET_VM: NOVALUE,
101     0197 1 DO_RECORD: NOVALUE, ! Process an include or omit
102     0198 1 GET_DEFN, ! Get position, size and type from factor
103     0199 1 GET_FFLD, ! Create a new forced field block
104     0200 1 GET_LINE, ! Get line of old spec file
105     0201 1 TREE_INSERT,
106     0202 1 SOR_ERROR, ! Issue diagnostic
107     0203 1 PUT_LITE: NOVALUE, ! Output a literal to new spec file
108     0204 1 PUT_QUOT: NOVALUE, ! Output quoted text to new spec file
109     0205 1 PUT_TEXT: NOVALUE CALL, ! Output text to new spec file
110     0206 1 PUT_LINE: NOVALUE CALL, ! Output line to new spec file
111     0207 1 NEW_KEY: CALL, ! Create a new key description
112     0208 1 UPCASE: NOVALUE, ! Convert to upper case
113     0209 1 FREE_FIXED: NOVALUE, ! Convert from free-format to fixed-format
114     0210 1 CLOSE_FILES: NOVALUE; ! Close input and output files
115     0211 1
116     0212 1 LITERAL
117     0213 1 HEADER_LINE = 1, ! Types of lines in old spec file
118     0214 1 ALTSEQ_LINE = 2,
119     0215 1 RECORD_LINE = 3,
120     0216 1 FIELD_LINE = 4;
121     0217 1
```

```
122 0218 1 LITERAL
123 0219 1 FALSE = 0;
124 0220 1 TRUE = 1;
125 0221 1
126 0222 1 STRUCTURE
127 0223 1 BBLOCK[O,P,S,E;BS=0] = [BS](BBLOCK+O)<P,S,E>;
128 0224 1
129 0225 1 MACRO
130 0226 1 BASE_ = 0,0,0,0 %,
131 0227 1 ELIF_ = ELSE IF %,
132 0228 1 LENADR_[] = %CHARCOUNT(%REMAINING), UPLIT BYTE(%STRING(%REMAINING)) %;
133 0229 1
134 L 0230 1 %IF %BLISS(BLISS32)
135 0231 1 %THEN MACRO STR_LOG_WORKFILE = 'SORTDO' %;
136 U 0232 1 %ELSE MACRO STR_LOG_WORKFILE = 'SYS$SCRATCH:SRTRN.SRT' %;
137 0233 1 %FI
138 0234 1
139 0235 1 LITERAL
140 0236 1 INP_BUF_WIDTH = 132,
141 0237 1 OUT_BUF_WIDTH = 255,
142 0238 1 OUT_WIDTH = 80 - 8 - 1;
143 0239 1
144 0240 1 OWN
145 0241 1 INP_BUF: VECTOR[INP_BUF_WIDTH,BYTE], ! Input record buffer
146 0242 1 OUT_BUF: VECTOR[OUT_BUF_WIDTH,BYTE], ! Output record buffer
147 0243 1 OUT_DSC: VECTOR[2], ! Len/adr of remaining room
148 0244 1 INP_REC_SIZ; ! Input record size
149 0245 1
150 0246 1 OWN
151 0247 1 CUR_RAB: REF $RAB_DECL; ! Current input RAB
152 0248 1 OWN
153 0249 1 INP_FNA: BLOCK[NAM$C_MAXRSS, BYTE], ! File name string area
154 P 0250 1 INP_NAM: $NAM(
155 P 0251 1 ESS=%ALLOCATION(INP_FNA), ! Expanded name string size
156 P 0252 1 ESA=INP_FNA, ! Expanded name string area
157 P 0253 1 RSS=%ALLOCATION(INP_FNA), ! Resultant name string size
158 0254 1 RSA=INP_FNA), ! Resultant name string area
159 P 0255 1 INP_FAB: $FAB(
160 P 0256 1 NAM=INP_NAM[BASE_],
161 P 0257 1 DNA=UPLIT BYTE('SRT'), ! Default for /S11 only!
162 P 0258 1 DNS=%CHARCOUNT('SRT'), ! Default for /S11 only!
163 P 0259 1 FAC=GET, ! Read-only
164 P 0260 1 FOP=<SQ0>, ! Sequential only
165 P 0261 1 RFM=VAR, ! Variable length format
166 0262 1 SHR=GET, ! Share read
167 P 0263 1 INP_RAB: $RAB(
168 P 0264 1 FAB=INP_FAB[BASE_],
169 P 0265 1 MBC=16, ! Multi-block count
170 P 0266 1 MBF=2, ! Multi-buffer count
171 P 0267 1 USZ=%ALLOCATION(INP_BUF),
172 0268 1 UBF=INP_BUF); ! Input buffer
173 0269 1 OWN
174 0270 1 TMP_FNA: BLOCK[NAM$C_MAXRSS, BYTE], ! File name string area
175 P 0271 1 TMP_NAM: $NAM(
176 P 0272 1 ESS=%ALLOCATION(TMP_FNA), ! Expanded name string size
177 P 0273 1 ESA=TMP_FNA, ! Expanded name string area
178 P 0274 1 RSS=%ALLOCATION(TMP_FNA), ! Resultant name string size
```



```
179      0275 1      RSA=TMP_FNA),          ! Resultant name string area
180      P 0276 1      TMP_FAB: $FAB(
181      P 0277 1      NAM=TMP_NAM(BASE ),
182      P 0278 1      FNA=UPLIT BYTE(STR_LOG_WORKFILE),
183      P 0279 1      FNS=%CHARCOUNT(STR_LOG_WORKFILE),
184      P 0280 1      FAC=<GET,PUT>,
185      P 0281 1      FOP=<DFW,DLT,TMD,TMP,SUP,SQO>,
186      P 0282 1      RFM=VAR,          ! Variable length format
187      P 0283 1      SHR=NIL,          ! No sharing
188      P 0284 1      TMP_RAB: $RAB(
189      P 0285 1      FAB=TMP_FAB(BASE ),
190      P 0286 1      MBC=16,          ! Multi-block count
191      P 0287 1      MBF=2,          ! Multi-buffer count
192      P 0288 1      USZ=%ALLOCATION(INP_BUF),          ! Same as input file
193      P 0289 1      UBF=INP_BUF,
194      P 0290 1      RBF=INP_BUF);          ! Output buffer
195      0291 1      OWN
196      P 0292 1      OUT_FNA: BLOCK[NAM$C_MAXRSS, BYTE],          ! File name string area
197      P 0293 1      OUT_NAM: $NAM(
198      P 0294 1      RLF=INP_NAM(BASE ),          ! Related file name string
199      P 0295 1      ESS=%ALLOCATION(OUT_FNA),          ! Expanded name string size
200      P 0296 1      ESA=OUT_FNA,          ! Expanded name string area
201      P 0297 1      RSS=%ALLOCATION(OUT_FNA),          ! Resultant name string size
202      P 0298 1      RSA=OUT_FNA),          ! Resultant name string area
203      P 0299 1      OUT_FAB: $FAB(
204      P 0300 1      NAM=OUT_NAM(BASE ),
205      P 0301 1      DNA=UPLIT BYTE('SRT'),
206      P 0302 1      DNS=%CHARCOUNT('SRT'),
207      P 0303 1      FOP=<OFP,SQO>,          ! Output file parse
208      P 0304 1      FAC=PUT,          ! Write access
209      P 0305 1      RAT=CR,          ! Line feed and carriage return
210      P 0306 1      RFM=VAR),          ! Variable length format
211      P 0307 1      OUT_RAB: $RAB(
212      P 0308 1      FAB=OUT_FAB(BASE ),
213      P 0309 1      MBC=16,          ! Multi-block count
214      P 0310 1      MBF=2,          ! Multi-buffer count
215      P 0311 1      RSZ=%ALLOCATION(OUT_BUF),
216      P 0312 1      RBF=OUT_BUF);          ! Output buffer
217      0313 1
218      0314 1      LITERAL
219      0315 1      LUN_K_MSG = 1,
220      0316 1      LUN_K_INP = 2,
221      0317 1      LUN_K_OUT = 3,
222      0318 1      LUN_K_TMP = 4;
223      0319 1
224      0320 1      ! Define shared messages
225      0321 1      !
226      0322 1      MACRO
227      M 0323 1      DEF SHR [MSG,SEV] =
228      M 0324 1      %NAME('SRTRN$',MSG) =
229      M 0325 1      %NAME('SHR$',MSG) + %NAME('STSSK$',SEV)
230      M 0326 1      %IF %BLISS(BLISS32) AND K NATIVE %THEN
231      M 0327 1      + SRTRN$_FACILITY ^ T6
232      M 0328 1      %FI
233      0329 1      %;
234      0330 1      LITERAL
235      P 0331 1      DEF SHR_(
```



```
236 P 0332 1 BADLOGIC, SEVERE, ! Internal logic error detected
237 P 0333 1 CLOSEDEL, ERROR, ! Error closing !AS
238 P 0334 1 CLOSEIN, ERROR, ! Error closing !AS as input
239 P 0335 1 CLOSEOUT, ERROR, ! Error closing !AS as output
240 P 0336 1 INSVIRMEM, SEVERE, ! Insufficient virtual memory
241 P 0337 1 OPENIN, SEVERE, ! Error opening !AS as input
242 P 0338 1 OPENOUT, SEVERE, ! Error opening !AS as output
243 P 0339 1 READERR, ERROR, ! Error reading !AS
244 P 0340 1 SYSERROR, SEVERE, ! System service error
245 P 0341 1 WRITEERR, ERROR); ! Error writing !AS
246 0342 1
247 0343 1 ! Define the keyword strings
248 0344 1
249 0345 1 MACRO
250 0346 1 DEFSTR_[A,B] = %NAME('X_',A) = B %QUOTE % %;
251 0347 1 MACRO
252 0348 1 DEFSTR_(KEYWORDS);
253 0349 1
254 0350 1
255 0351 1 ! Define a structure for holding field definitions
256 0352 1
257 0353 1 $UNIT_FIELD
258 0354 1 FLD_FIELDS =
259 0355 1 SET
260 0356 1 FLD_LSON= [ $ADDRESS], ! Pointer to FLD_BLOCK
261 0357 1 FLD_RSON= [ $ADDRESS], ! Pointer to FLD_BLOCK
262 0358 1 FLD_POS= [ $BITS(16)], ! Position of field in record
263 0359 1 FLD_SIZE= [ $BITS(16)], ! Size of field in record
264 0360 1 FLD_FFLD= [ $ADDRESS], ! Pointer to FFLD_BLOCK
265 0361 1 FLD_DTY= [ $BITS(8)] ! Datatype of field
266 0362 1 TES;
267 0363 1 LITERAL FLD_K_SIZE= $FIELD SET UNITS; ! Size in units
268 0364 1 MACRO FLD_BLOCK= BBLOCK[FLD_K_SIZE] FIELD(FLD_FIELDS) %;
269 0365 1
270 0366 1 ! Define a structure for holding forced field definitions
271 0367 1
272 0368 1 $UNIT_FIELD
273 0369 1 FFLD_FIELDS =
274 0370 1 SET
275 0371 1 FFLD_NEXT= [ $ADDRESS], ! Pointer to alternate FFLD_BLOCK or 0
276 0372 1 FFLD_COND= [ $BITS(16)], ! Condition number
277 0373 1 FFLD_LIT= [ $BITS(8)] ! Literal value for this FFLD
278 0374 1 TES;
279 0375 1 LITERAL FFLD_K_SIZE= $FIELD SET UNITS; ! Size in units
280 0376 1 MACRO FFLD_BLOCK= BBLOCK[FFLD_K_SIZE] FIELD(FFLD_FIELDS) %;
281 0377 1
282 0378 1 ! Define a structure for format information
283 0379 1
284 0380 1 $UNIT_FIELD
285 0381 1 FMT_FIELDS =
286 0382 1 SET
287 0383 1 FMT_NEXT= [ $ADDRESS], ! Pointer to next FMT_BLOCK
288 0384 1 FMT_KEYS= [ $ADDRESS], ! Pointer to KEY_BLOCK
289 0385 1 FMT_CKEY= [ $ADDRESS], ! Pointer to current KEY_BLOCK
290 0386 1 FMT_OUTLRL= [ $INTEGER], ! Output LRL for this format
291 0387 1 FMT_COND= [ $BITS(16)] ! Conditionality of this format
292 0388 1 TES;
```



```
293 0389 1 LITERAL FMT_K_SIZE= $FIELD SET UNITS; ! Size in units
294 0390 1 MACRO FMT_BLOCK= BBLOCK[FMT_K_SIZE] FIELD(FMT_FIELDS) %;
295 0391 1
296 0392 1 ! Define a structure for holding keys of different formats
297 0393 1
298 0394 1 $UNIT_FIELD
299 0395 1 KEY_FIELDS =
300 0396 1 SET
301 0397 1 KEY_NEXT= [$ADDRESS], ! Pointer to next KEY_BLOCK
302 0398 1 KEY_FLD= [$ADDRESS], ! Pointer to FLD_BLOCK
303 0399 1 KEY_ASCE= [$BIT], ! True if ascending key
304 0400 1 $OVERLAY(KEY_ASCE)
305 0401 1 KEY_LIT= [$BITS(8)] ! Literal key (if key_fld = 0)
306 0402 1 $CONTINUE
307 0403 1 TES;
308 0404 1 LITERAL KEY_K_SIZE= $FIELD SET UNITS; ! Size in units
309 0405 1 MACRO KEY_BLOCK= BBLOCK[KEY_K_SIZE] FIELD(KEY_FIELDS) %;
310 0406 1
311 0407 1
312 0408 1 ! OWN Storage
313 0409 1
314 0410 1 GLOBAL
315 0411 1 CLR 0: VECTOR[0],
316 0412 1 ASCENDING, ! True if ascending is the default
317 0413 1 ALTSEQ, ! Altseq expected = 1, unexpected = 0
318 0414 1 FLD_ROOT, ! Tree of field names
319 0415 1 FMT_ROOT: REF FMT_BLOCK,
320 0416 1 FMT_CURR: REF FMT_BLOCK,
321 0417 1 LINE_TYPE, ! Type of input line
322 0418 1 LINE_NUM, ! Last line number
323 0419 1 LINE_SEQ, ! True if line numbers were out of sequence
324 0420 1 ADDINKEY, ! True if additional keys were added
325 0421 1 COND_NUM, ! Condition number
326 0422 1 PASS, ! What pass we are working on
327 0423 1 PROCESS, ! What sort process we are doing
328 0424 1 EBCDIC, ! True for ebcidic collating sequence
329 0425 1 NOSTRIP, ! True if we are not doing key stripping
330 0426 1 OUTLRL, ! Output LRL
331 0427 1 SWITCH S32, ! True if /S32 qualifier was present
332 0428 1 SOR_SEV, ! Worst severity (encoded)
333 0429 1 SOR_STS, ! Worst status
334 0430 1 CLR_1: VECTOR[0];
335 0431 1
336 U 0432 1 %IF %BLISS(BLISS16) %THEN
337 U 0433 1 EXTERNAL LITERAL
338 U 0434 1 EX$SEV, EX$SUC, EX$WAR, EX$ERR;
339 0435 1 %FI
340 0436 1 %IF NOT %DECLARED(DSC$K_DTYPE_DIBOL) %THEN
341 0437 1 LITERAL
342 0438 1 DSC$K_DTYPE_DIBOL = 40;
343 0439 1 %FI
344 0440 1 %IF NOT %DECLARED(DSC$K_DTYPE_AZ) %THEN
345 0441 1 LITERAL
346 0442 1 DSC$K_DTYPE_AZ = 41;
347 0443 1 %FI
```

```
0444 1 GLOBAL ROUTINE SOR_ERROR(ERR) =
0445 1
0446 1 ++
0447 1
0448 1 FUNCTIONAL DESCRIPTION:
0449 1
0450 1     This routine signals an error diagnostic.
0451 1
0452 1 FORMAL PARAMETERS:
0453 1
0454 1     Parameters passed to SIGNAL.
0455 1
0456 1 IMPLICIT INPUTS:
0457 1
0458 1     NONE
0459 1
0460 1 IMPLICIT OUTPUTS:
0461 1
0462 1     NONE
0463 1
0464 1 ROUTINE VALUE:
0465 1
0466 1     System status (first parameter of signalled status), with the
0467 1     INHIB_MSG bit set.
0468 1
0469 1 SIDE EFFECTS:
0470 1
0471 1     The image may be exited due to the error.
0472 1
0473 1 --
0474 2 BEGIN
0475 2 SIGNAL(.ERR);
0476 2 RETURN .ERR OR STSSM_INHIB_MSG;
0477 1 END;
```

```
                                .TITLE  SRTTRN
                                .IDENT   \V04-000\
                                .PSECT    $PLITS$,NOWRT,NOEXE,2
30  44  54  52  53  2E  00000 P.AAA:  .ASCII  \.SRT\
                                .ASCII  \SORTDO\
                                .ASCII  \.SRT\
                                .PSECT    $OWNS$,NOEXE,2
                                00000 INP_BUF: .BLKB  132
                                00084 OUT_BUF: .BLKB  255
                                00183         .BLKB   1
                                00184 OUT_DSC: .BLKB   8
                                0018C INP_REC_SIZ:
                                        .BLKB   4
                                00190 CUR_RAB: .BLKB   4
                                00194 INP_FNA: .BLKB  255
                                00293         .BLKB   1
02  00294 INP_NAM: .BYTE  2
```


Address	Hex	Label	Value
60	00295	.BYTE	96
FF	00296	.BYTE	-1
00	00297	.BYTE	0
00000000	00298	.ADDRESS	INP_FNA
00	0029C	.BYTE	0
00	0029D	.BYTE	0
FF	0029E	.BYTE	-1
00	0029F	.BYTE	0
00000000	002A0	.ADDRESS	INP_FNA
00000000	002A4	.LONG	0
0000#	002A8	.WORD	0[8]
0000#	002B8	.WORD	0[3]
0000#	002BE	.WORD	0[3]
00000000	002C4	.LONG	0
00000000	002C8	.LONG	0
00	002CC	.BYTE	0
00	002CD	.BYTE	0
00	002CE	.BYTE	0
00	002CF	.BYTE	0
00	002D0	.BYTE	0
00	002D1	.BYTE	0
00#	002D2	.BYTE	0[2]
00000000	002D4	.LONG	0
00000000	002D8	.LONG	0
00000000	002DC	.LONG	0
00000000	002E0	.LONG	0
00000000	002E4	.LONG	0
00000000	002E8	.LONG	0
00000000#	002EC	.LONG	0[2]
03	002F4	INP_FAB: .BYTE	3
50	002F5	.BYTE	80
0000	002F6	.WORD	0
00000040	002F8	.LONG	64
00000000	002FC	.LONG	0
00000000	00300	.LONG	0
00000000	00304	.LONG	0
0000	00308	.WORD	0
02	0030A	.BYTE	2
02	0030B	.BYTE	2
00000000	0030C	.LONG	0
00	00310	.BYTE	0
00	00311	.BYTE	0
00	00312	.BYTE	0
02	00313	.BYTE	2
00000000	00314	.LONG	0
00000000	00318	.LONG	0
00000000	0031C	.ADDRESS	INP_NAM
00000000	00320	.LONG	0
00000000	00324	.ADDRESS	P.AAA
00	00328	.BYTE	0
04	00329	.BYTE	4
0000	0032A	.WORD	0
00000000	0032C	.LONG	0
0000	00330	.WORD	0
00	00332	.BYTE	0
00	00333	.BYTE	0
00000000	00334	.LONG	0

```
00000000 00338 .LONG 0
0000 0033C .WORD 0
00 0033E .BYTE 0
00 0033F .BYTE 0
00000000 00340 .LONG 0
01 00344 INP_RAB: .BYTE 1
44 00345 .BYTE 68
0000 00346 .WORD 0
00000000 00348 .LONG 0
00000000 0034C .LONG 0
00000000 00350 .LONG 0
0000# 00354 .WORD 0[3]
0000 0035A .WORD 0
00000000 0035C .LONG 0
0000 00360 .WORD 0
00 00362 .BYTE 0
00 00363 .BYTE 0
0084 00364 .WORD 132
0000 00366 .WORD 0
00000000' 00368 .ADDRESS INP_BUF
00000000 0036C .LONG 0
00000000 00370 .LONG 0
00000000 00374 .LONG 0
00 00378 .BYTE 0
00 00379 .BYTE 0
00 0037A .BYTE 0
00 0037B .BYTE 0
00000000 0037C .LONG 0
00000000' 00380 .ADDRESS INP_FAB
00000000 00384 .LONG 0
00388 TMP_FNA: .BLKB 255
00487 .BLKB 1
02 00488 TMP_NAM: .BYTE 2
60 00489 .BYTE 96
FF 0048A .BYTE -1
00 0048B .BYTE 0
00000000' 0048C .ADDRESS TMP_FNA
00 00490 .BYTE 0
00 00491 .BYTE 0
FF 00492 .BYTE -1
00 00493 .BYTE 0
00000000' 00494 .ADDRESS TMP_FNA
00000000 00498 .LONG 0
0000# 0049C .WORD 0[8]
0000# 004AC .WORD 0[3]
0000# 004B2 .WORD 0[3]
00000000 004B8 .LONG 0
00000000 004BC .LONG 0
00 004C0 .BYTE 0
00 004C1 .BYTE 0
00 004C2 .BYTE 0
00 004C3 .BYTE 0
00 004C4 .BYTE 0
00 004C5 .BYTE 0
00# 004C6 .BYTE 0[2]
00000000 004C8 .LONG 0
00000000 004CC .LONG 0
```



```
00000000 004D0 .LONG 0
00000000 004D4 .LONG 0
00000000 004D8 .LONG 0
00000000 004DC .LONG 0
00000000# 004E0 .LONG 0[2]
    03 004E8 TMP_FAB: .BYTE 3
    50 004E9 .BYTE 80
    0000 004EA .WORD 0
0000807C 004EC .LONG 32892
00000000 004F0 .LONG 0
00000000 004F4 .LONG 0
00000000 004F8 .LONG 0
    0000 004FC .WORD 0
    03 004FE .BYTE 3
    20 004FF .BYTE 32
00000000 00500 .LONG 0
    00 00504 .BYTE 0
    00 00505 .BYTE 0
    00 00506 .BYTE 0
    02 00507 .BYTE 2
00000000 00508 .LONG 0
00000000 0050C .LONG 0
00000000' 00510 .ADDRESS TMP_NAM
00000000' 00514 .ADDRESS P.AAB
00000000 00518 .LONG 0
    06 0051C .BYTE 6
    00 0051D .BYTE 0
    0000 0051E .WORD 0
00000000 00520 .LONG 0
    0000 00524 .WORD 0
    00 00526 .BYTE 0
    00 00527 .BYTE 0
00000000 00528 .LONG 0
00000000 0052C .LONG 0
    0000 00530 .WORD 0
    00 00532 .BYTE 0
    00 00533 .BYTE 0
00000000 00534 .LONG 0
    01 00538 TMP_RAB: .BYTE 1
    44 00539 .BYTE 68
    0000 0053A .WORD 0
00000000 0053C .LONG 0
00000000 00540 .LONG 0
00000000 00544 .LONG 0
    0000# 00548 .WORD 0[3]
    0000 0054E .WORD 0
00000000 00550 .LONG 0
    0000 00554 .WORD 0
    00 00556 .BYTE 0
    00 00557 .BYTE 0
    0084 00558 .WORD 132
    0000 0055A .WORD 0
00000000' 0055C .ADDRESS INP_BUF
00000000' 00560 .ADDRESS INP_BUF
00000000 00564 .LONG 0
00000000 00568 .LONG 0
    00 0056C .BYTE 0
```

```
00 0056D .BYTE 0
00 0056E .BYTE 0
00 0056F .BYTE 0
00000000 00570 .LONG 0
00000000 00574 .ADDRESS TMP_FAB
00000000 00578 .LONG 0
0057C OUT_FNA: .BLKB 255
0057B .BLKB 1
02 0067C OUT_NAM: .BYTE 2
60 0067D .BYTE 96
FF 0067E .BYTE -1
00 0067F .BYTE 0
00000000 00680 .ADDRESS OUT_FNA
00 00684 .BYTE 0
00 00685 .BYTE 0
FF 00686 .BYTE -1
00 00687 .BYTE 0
00000000 00688 .ADDRESS OUT_FNA
00000000 0068C .ADDRESS INP_NAM
0000# 00690 .WORD 0[8]
0000# 006A0 .WORD 0[3]
0000# 006A6 .WORD 0[3]
00000000 006AC .LONG 0
00000000 006B0 .LONG 0
00 006B4 .BYTE 0
00 006B5 .BYTE 0
00 006B6 .BYTE 0
00 006B7 .BYTE 0
00 006B8 .BYTE 0
00 006B9 .BYTE 0
00# 006BA .BYTE 0[2]
00000000 006BC .LONG 0
00000000 006C0 .LONG 0
00000000 006C4 .LONG 0
00000000 006C8 .LONG 0
00000000 006CC .LONG 0
00000000 006D0 .LONG 0
00000000# 006D4 .LONG 0[2]
03 006DC OUT_FAB: .BYTE 3
50 006DD .BYTE 80
0000 006DE .WORD 0
20000040 006E0 .LONG 536870976
00000000 006E4 .LONG 0
00000000 006E8 .LONG 0
00000000 006EC .LONG 0
0000 006F0 .WORD 0
01 006F2 .BYTE 1
00 006F3 .BYTE 0
00000000 006F4 .LONG 0
00 006F8 .BYTE 0
00 006F9 .BYTE 0
02 006FA .BYTE 2
02 006FB .BYTE 2
00000000 006FC .LONG 0
00000000 00700 .LONG 0
00000000 00704 .ADDRESS OUT_NAM
00000000 00708 .LONG 0
```



```

00000000' 0070C .ADDRESS P.AAC
      00 00710 .BYTE 0
      04 00711 .BYTE 4
      0000 00712 .WORD 0
00000000 00714 .LONG 0
      0000 00718 .WORD 0
      00 0071A .BYTE 0
      00 0071B .BYTE 0
00000000 0071C .LONG 0
00000000 00720 .LONG 0
      0000 00724 .WORD 0
      00 00726 .BYTE 0
      00 00727 .BYTE 0
00000000 00728 .LONG 0
      01 0072C OUT_RAB: .BYTE 1
      44 0072D .BYTE 68
      0000 0072E .WORD 0
00000000 00730 .LONG 0
00000000 00734 .LONG 0
00000000 00738 .LONG 0
      0000# 0073C .WORD 0[3]
      0000 00742 .WORD 0
00000000 00744 .LONG 0
      0000 00748 .WORD 0
      00 0074A .BYTE 0
      00 0074B .BYTE 0
      0000 0074C .WORD 0
      00FF 0074E .WORD 255
00000000 00750 .LONG 0
00000000' 00754 .ADDRESS OUT_BUF
00000000 00758 .LONG 0
0000C000 0075C .LONG 0
      00 00760 .BYTE 0
      00 00761 .BYTE 0
      00 00762 .BYTE 0
      00 00763 .BYTE 0
00000000 00764 .LONG 0
00000000' 00768 .ADDRESS OUT_FAB
00000000 0076C .LONG 0

```

.PSECT \$GLOBAL\$,NOEXE,2

```

00000 CLR_0:: .BLKB 0
00000 ASCENDING::
      .BLKB 4
00004 ALTSEQ:: .BLKB 4
00008 FLD_ROOT::
      .BLKB 4
0000C FMT_ROOT::
      .BLKB 4
00010 FMT_CURR::
      .BLKB 4
00014 LINE_TYPE::
      .BLKB 4
00018 LINE_NUM::
      .BLKB 4
0001C LINE_SEQ::

```

SRTTRN
V04-000

N 12
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 14
(3)

00020 ADDINCKEY:: .BLKB 4
00024 COND_NUM:: .BLKB 4
00028 PASS:: .BLKB 4
0002C PROCESS:: .BLKB 4
00030 EBCDIC:: .BLKB 4
00034 NOSTRIP:: .BLKB 4
00038 OUTLRL:: .BLKB 4
0003C SWITCH_S32:: .BLKB 4
00040 SOR_SEV:: .BLKB 4
00044 SOR_STS:: .BLKB 4
00048 CLR_1:: .BLKB 0

0000 00000
04 AC DD 00002
00000000G 00 01 FB 00005
50 04 AC 10000000 8F C9 0000C
04 00015

.PSECT \$CODE\$,NOWRT,2

.ENTRY SOR_ERROR, Save nothing
PUSHL ERR
CALLS #1, LIB\$SIGNAL
BISL3 #268435456, ERR, R0
RET

: 0444
: 0475
: 0476
: 0477

; Routine Size: 22 bytes, Routine Base: \$CODE\$ + 0000


```
: 384      0478 1 ROUTINE CVTBTA(X, P) =  
: 385      0479 2 BEGIN  
: 386      0480 2 IF .X LSS 0  
: 387      0481 2 THEN  
: 388      0482 3 BEGIN  
: 389      0483 3 CH$WCHAR_A('-', P);  
: 390      0484 3 X = -.X;  
: 391      0485 2 END;  
: 392      0486 2 IF .X GEQ 10 THEN P = CVTBTA(.X/10, .P);  
: 393      0487 2 CH$WCHAR_A((.X MOD 10)+'0', P);  
: 394      0488 2 RETURN .P;  
: 395      0489 1 END;
```

				0000 00000 CVTBTA: .WORD	Save nothing		0478
		04	AC	D5 00002	TSTL	X	0480
			0C	18 00005	BGEQ	1\$	
	08	BC		2D 90 00007	MOVB	#45, @P	0483
			08	AC D6 0000B	INCL	P	
	04	AC	04	AC CE 0000E	MNEGL	X, X	0484
		0A	04	AC D1 00013	CMPL	X, #10	0486
				10 19 00017	BLSS	2\$	
			08	AC DD 00019	PUSHL	P	
	7E			0A C7 0001C	DIVL3	#10, X, -(SP)	
		04	DB	AF 00021	CALLS	#2, CVTBTA	
		08	AC	50 D0 00025	MOVL	R0, P	
7E		00	04	AC 01 7A 00029	EMUL	#1, X, #0, -(SP)	0487
50		50		0A 7B 0002F	EDIV	#10, (SP)+, R0, R0	
	08	BC		50 30 81 00034	ADDB3	#48, R0, @P	
			08	AC D6 00039	INCL	P	
			08	AC D0 0003C	MOVL	P, R0	0488
				04 00040	RET		0489

; Routine Size: 65 bytes, Routine Base: \$CODE\$ + 0016


```

: 397 L 0490 1 %IF %BLISS(BLISS32) AND K_NATIVE
: 398 U 0491 1 %THEN %ELSE
: 399 U 0492 1 ROUTINE PUT_MSG(LEN, ADR): NOVALUE =
: 400 U 0493 1 BEGIN
: 401 U 0494 1 %IF %BLISS(BLISS32)
: 402 U 0495 1 %THEN
: 403 U 0496 1 $XPO PUT MSG(
: 404 U 0497 1 STRING = (.LEN, .ADR),
: 405 U 0498 1 FAILURE = 0);
: 406 U 0499 1 %ELSE
: 407 U 0500 1 OWN
: 408 U 0501 1 LUN: INITIAL(0);
: 409 U 0502 1 EXTERNAL
: 410 U 0503 1 $DSW; ! Directive status word
: 411 U 0504 1 LOCAL
: 412 U 0505 1 IOSB: VECTOR[4,BYTE]; ! I/O status block
: 413 U 0506 1
: 414 U 0507 1 IF .LUN EQL 0
: 415 U 0508 1 THEN
: 416 U 0509 1 BEGIN
: 417 U 0510 1 |
: 418 U 0511 1 | Assign lun 1
: 419 U 0512 1 |
: 420 U 0513 1 | ALUN$$ ( LUN_K_MSG, 'TI', 0 );
: 421 U 0514 1 | IF .SDSW NEQ IS_SUC THEN RETURN;
: 422 U 0515 1 | LUN = LUN_K_MSG;
: 423 U 0516 1 | END;
: 424 U 0517 1
: 425 U 0518 1 | Attach the device and write the buffer
: 426 U 0519 1 |
: 427 U 0520 1 | QIOW$$ ( IO_ATT, LUN_K_MSG, 10, IOSB );
: 428 U 0521 1 | IF .IOSB[0] NEQ IS_SUC THEN RETURN;
: 429 U 0522 1 | QIOW$$ ( IO_WLB, LUN_K_MSG, 10, IOSB, <.ADR, .LEN> );
: 430 U 0523 1 | IF .IOSB[0] NEQ IS_SUC THEN RETURN;
: 431 U 0524 1
: 432 U 0525 1 | Detatch the device
: 433 U 0526 1 |
: 434 U 0527 1 | QIOW$$ ( IO_DET, LUN_K_MSG, 10, IOSB );
: 435 U 0528 1 | IF .IOSB[0] NEQ IS_SUC THEN RETURN;
: 436 U 0529 1 %FI
: 437 U 0530 1 END;
: 438 U 0531 1 %FI
```



```

: 440      0532 1 ROUTINE COND_HAND(
: 441      0533 1   SIG: REF VECTOR,
: 442      0534 1   MCH: REF VECTOR,
: 443      0535 1   ENA: REF VECTOR) =
: 444      0536 2   BEGIN
: 445      0537 2   BIND
: 446      0538 2   COND = SIG[1]: %BLISS32(BLOCK[BYTE]) %BLISS16(CONDITION_VALUE),
: 447      0539 2   RETURN_VALUE = MCH[ %BLISS16(1) %BLISS32(3) ];
: 448      0540 2   LITERAL
: 449      0541 2   FAO_COUNT = 2, ! Index in SIG of the FAO count
: 450      0542 2   SIG_IGNORE = %IF %BLISS(BLISS32) %THEN 2 %ELSE 0 %FI;
: 451      0543 2
: 452      L 0544 2 %IF %BLISS(BLISS32) AND K_NATIVE
: 453      0545 2 %THEN
: 454      0546 2   BEGIN
: 455      0547 2   RETURN SS$_RESIGNAL;
: 456      0548 2   END
: 457      U 0549 2 %ELSE
: 458      UU 0550 2   BEGIN
: 459      UU 0551 2   LITERAL
: 460      UU 0552 2   BUF_SIZE = 255;
: 461      UU 0553 2   %IF %BLISS(BLISS32) %THEN LOCAL %ELSE OWN %FI
: 462      UU 0554 2   B: VECTOR[BUF_SIZE,BYTE];
: 463      UU 0555 2   LOCAL
: 464      UU 0556 2   P,
: 465      UU 0557 2   Q: REF VECTOR[BYTE,UNSIGNED],
: 466      UU 0558 2   X;
: 467      UU 0559 2   BIND
: 468      UU 0560 2   SEV = UPLIT('WSEIF567'): VECTOR[BYTE];
: 469      UU 0561 2   MACRO
: 470      UU 0562 2   CRLF = %CHAR(%X'0D',%X'0A') %;
: 471      UU 0563 2
: 472      UU 0564 2   ! Issue the original error message
: 473      UU 0565 2   !
: 474      UU 0566 2   P = CH$MOVE(LENADR (%STRING(CRLF,%SRTTRN-')), B[0]);
: 475      UU 0567 2   CH$WCHAR_A(.SEV[COND[ST$V_SEVERITY]], P);
: 476      UU 0568 2   CH$WCHAR_A('-', P);
: 477      UU 0569 2
: 478      UU 0570 2   ! Is this a known message?
: 479      UU 0571 2   !
: 480      UU 0572 2   IF .COND[ST$V_COND_ID] GEQU .ERR_DC[-1]
: 481      UU 0573 2   THEN
: 482      UU 0574 2   Q = UPLIT BYTE(%ASCIC 'NOMSG', %ASCIC 'Message number ')
: 483      UU 0575 2   ELSE
: 484      UU 0576 2   Q = .ERR_DC[COND[ST$V_COND_ID]];
: 485      UU 0577 2   P = CH$MOVE(-Q[0], Q[1], .P);
: 486      UU 0578 2   Q = .Q + .Q[0] + 1;
: 487      UU 0579 2   CH$WCHAR_A(' ', P);
: 488      UU 0580 2   CH$WCHAR_A(' ', P);
: 489      UU 0581 2
: 490      UU 0582 2   ! Index of first (if any) FAO parameter
: 491      UU 0583 2   !
: 492      UU 0584 2   X = FAO_COUNT+1;
: 493      UU 0585 2
: 494      UU 0586 2   ! Look for and substitute any !AS in the control string
: 495      UU 0587 2   !
: 496      U 0588 2   BEGIN
```

497 U 0589
498 U 0590
499 U 0591
500 U 0592
501 U 0593
502 U 0594
503 U 0595
504 U 0596
505 U 0597
506 U 0598
507 U 0599
508 U 0600
509 U 0601
510 U 0602
511 U 0603
512 U 0604
513 U 0605
514 U 0606
515 U 0607
516 U 0608
517 U 0609
518 U 0610
519 U 0611
520 U 0612
521 U 0613
522 U 0614
523 U 0615
524 U 0616
525 U 0617
526 U 0618
527 U 0619
528 U 0620
529 U 0621
530 U 0622
531 U 0623
532 U 0624
533 U 0625
534 U 0626
535 U 0627
536 U 0628
537 U 0629
538 U 0630
539 U 0631
540 U 0632
541 U 0633
542 U 0634
543 U 0635
544 U 0636
545 U 0637
546 U 0638
547 U 0639
548 U 0640
549 U 0641
550 U 0642
551 U 0643
552 U 0644
553 U 0645

```
LOCAL D: VECTOR[2];
MACRO FAO_STR = '!AS' %;
D[0] = .Q[0];
D[1] = .Q[1];
WHILE .D[0] GTR 0 DO
  BEGIN
    LOCAL R;
    R = CH$FIND_SUB(.D[0], .D[1], LENADR(FAO_STR));
    IF CH$FAIL(.R) THEN R = .D[0] ELSE R = CH$DIFF(.R, .D[1]);
    R = MINU(.R, CH$DIFF(B[BUF_SIZE], .P));
    P = CH$MOVE(.R, .D[1], .P);
    R = .R + %CHARCOUNT(FAO_STR);
    D[0] = .D[0] - .R;
    D[1] = .D[1] + .R;
    IF .X LEQ .SIG[0] - SIG_IGNORE AND
        .X LEQ .SIG[FAO_COUNT] + FAO_COUNT
    THEN
      BEGIN
        R = MINU(.VECTOR[SIG[X], 0], CH$DIFF(B[BUF_SIZE], .P));
        P = CH$MOVE(.R, .VECTOR[SIG[X], 1], .P);
        X = .X + 1;
      END;
    END;
  ! If this is an unknown message, append the number
  !
  IF .COND[ST$V_COND_ID] GEQU .ERR_D[-1]
  THEN
    P = CVTBTA(.COND, .P);
  END;
  ! Ensure that we've skipped all the FAO parameters
  !
  IF .SIG[0] GEQ FAO_COUNT - SIG_IGNORE
  THEN
    X = FAO_COUNT + 1 + .SIG[FAO_COUNT];
  ! Issue the first line
  !
  P = CH$MOVE(LENADR(CRLF), .P);
  PUT_MSG(CH$DIFF(.P, B[0]), B[0]);
  ! Print any other parameters as numbers
  !
  IF .X LEQ .SIG[0] - SIG_IGNORE
  THEN
    BEGIN
      P = CH$MOVE(LENADR('-I-ADDSTS, additional status '), B[0]);
      WHILE TRUE DO
        BEGIN
          IF CH$DIFF(B[BUF_SIZE], .P) LEQ 16 THEN EXITLOOP;
          P = CVTBTA(.SIG[X], .P);
          IF .X EQL .SIG[0] - SIG_IGNORE THEN EXITLOOP;
          P = CH$MOVE(LENADR(' /-'), .P);
          X = .X + 1;
        END;
      P = CH$MOVE(LENADR(CRLF), .P);
    END;
```



```
554      U 0646      PUT_MSG(CH$DIFF(.P,B[0]),B[0]);
555      U 0647      END;
556      U 0648
557      U 0649      ! Hang onto the worst error we've seen
558      U 0650      !
559      U 0651      BEGIN
560      U 0652      BIND CVT_SEV = UPLIT BYTE(2,0,3,1,4,5,6,7): VECTOR[BYTE];
561      U 0653      LOCAL SEV;
562      U 0654      SEV = .CVT_SEV[COND[STSSV_SEVERITY]];
563      U 0655      IF .SEV GTRU .SOR_SEV
564      U 0656      THEN
565      U 0657          %IF %BLISS(BLISS32)
566      U 0658          %THEN
567      U 0659              BEGIN
568      U 0660                  SOR_SEV = .SEV;
569      U 0661                  SOR_STS = .COND OR STSM_INHIB_MSG;
570      U 0662                  END
571      U 0663              %ELSE
572      U 0664                  BEGIN
573      U 0665                      BIND
574      U 0666                          CVT_STS = UPLIT(
575      U 0667                              EX$SUC, EX$SUC, EX$WAR, EX$ERR,
576      U 0668                              EX$SEV, EX$SEV, EX$SEV, EX$SEV): VECTOR;
577      U 0669                      SOR_SEV = .SEV;
578      U 0670                      SOR_STS = .CVT_STS[.SEV] OR STSM_INHIB_MSG;
579      U 0671                      END
580      U 0672                  %FI;
581      U 0673          END;
582      U 0674
583      U 0675      ! Abort if the severity is fatal (or worse)
584      U 0676      !
585      U 0677      %IF %BLISS(BLISS16)
586      U 0678      %THEN
587      U 0679          IF .SOR_SEV GEQU 4 THEN EXSTS( .SOR_STS );
588      U 0680      %FI
589      U 0681      RETURN 1;      ! Just resignal
590      U 0682      END
591      U 0683      %FI;
592      U 0684      END;
```

```
0000 0000 COND_HAND:
50      0918 8F 3C 00002      .WORD      Save nothing
                                MOVZWL      #2328, R0
                                04 00007      RET
; Routine Size: 8 bytes,      Routine Base: $CODE$ + 0057
```

```
: 0532
: 0547
: 0684
```

```
.. 594 0685 1 ROUTINE SRTTRN =
.. 595 0686 1
.. 596 0687 1 ++
.. 597 0688 1
.. 598 0689 1 FUNCTIONAL DESCRIPTION:
.. 599 0690 1
.. 600 0691 1 This is the main entry point for the spec file translator.
.. 601 0692 1
.. 602 0693 1 FORMAL PARAMETERS:
.. 603 0694 1
.. 604 0695 1 NONE
.. 605 0696 1
.. 606 0697 1 IMPLICIT INPUTS:
.. 607 0698 1
.. 608 0699 1 NONE
.. 609 0700 1
.. 610 0701 1 IMPLICIT OUTPUTS:
.. 611 0702 1
.. 612 0703 1 NONE
.. 613 0704 1
.. 614 0705 1 ROUTINE VALUE:
.. 615 0706 1
.. 616 0707 1 Status code.
.. 617 0708 1
.. 618 0709 1 SIDE EFFECTS:
.. 619 0710 1
.. 620 0711 1 NONE
.. 621 0712 1
.. 622 0713 1 --
.. 623 0714 2 BEGIN
.. 624 0715 2 LOCAL
.. 625 0716 2 PTR;
.. 626 0717 2
.. 627 0718 2 ENABLE
.. 628 0719 2 COND_HAND;
.. 629 0720 2
.. 630 0721 2 ! Initialize our storage
.. 631 0722 2 !
.. 632 0723 2 CH$FILL( 0, CLR_1[0] - CLR_0[0], CLR_0[0] );
.. 633 0724 2
.. 634 0725 2 ! Initialize the severity and message to success
.. 635 0726 2 !
.. 636 0727 2 SOR_SEV = 0;
.. 637 0728 2 SOR_STS = %BLISS32(SS$_NORMAL) %BLISS16(EX$$SUC);
.. 638 0729 2
.. 639 0730 2 OUT_DSC[0] = OUT_WIDTH;
.. 640 0731 2 OUT_DSC[1] = OUT_BUF[0];
.. 641 0732 2
.. 642 0733 2 INIT_IO();
.. 643 0734 2
.. 644 0735 2 ASCENDING = TRUE;
.. 645 0736 2
.. 646 0737 2 !+
.. 647 0738 2
.. 648 0739 2 Make a first pass to:
.. 649 0740 2 Output original file as comments
.. 650 0741 2 Process the header line
```



```
651 0742 2 | Define the fields
652 0743 2 |
653 0744 2 | Make a second pass to:
654 0745 2 | Describe the fields
655 0746 2 | Process the altseq line
656 0747 2 | Describe conditions
657 0748 2 |
658 0749 2 | Make a third pass to:
659 0750 2 | Describe omits/includes and keys
660 0751 2 |
661 0752 2 |
662 0753 2 | LINE_SEQ = FALSE;
663 0754 2 | EBCDIC = FALSE;
664 0755 2 | PROCESS = %C'R';
665 0756 2 | LINE_NUM = -1;
666 0757 2 |
667 0758 2 | PASS = 0;
668 0759 2 | WHILE TRUE DO
669 0760 2 | BEGIN
670 0761 2 |
671 0762 3 | PASS = .PASS + 1;
672 0763 3 |
673 0764 3 | SELECTONE .PASS OF
674 0765 3 | SET
675 0766 4 | [1]:BEGIN
676 0767 4 | PUT_TEXT(LENADR('!++'));
677 0768 4 | PUT_LINE();
678 0769 3 | END;
679 0770 4 | [2]:BEGIN
680 0771 4 | PUT_TEXT(LENADR('!--'));
681 0772 4 | PUT_LINE();
682 0773 4 | SELECTONE .PROCESS OF
683 0774 4 | SET
684 0775 4 | ['R']: PUT_TEXT(LENADR('/'X_PROC,'X_REC'));
685 0776 4 | ['T']: PUT_TEXT(LENADR('/'X_PROC,'X_TAG'));
686 0777 4 | ['A']: PUT_TEXT(LENADR('/'X_PROC,'X_ADDR'));
687 0778 4 | ['I']: PUT_TEXT(LENADR('/'X_PROC,'X_INDX'));
688 0779 4 | [OTHERWISE]: 0;
689 0780 4 | TES;
690 0781 4 | IF .EBCDIC
691 0782 4 | THEN
692 0783 4 | PUT_TEXT(LENADR('/'X_COLL,'('X_SEQU','X_EBC,')'));
693 0784 4 |
694 0785 4 | | Define all fields that will be needed
695 0786 4 | | Then actually issue text to describe them.
696 0787 4 | | Also, describe subfields that are used in key specifications.
697 0788 4 | |
698 0789 4 | DEF_SUBFIELDS();
699 0790 4 | DEF_FIELDS(.FLD_ROOT);
700 0791 3 | END;
701 0792 4 | [3]:BEGIN
702 0793 4 | 0;
703 0794 3 | END;
704 0795 3 | TES;
705 0796 3 |
706 0797 3 | | Initialize the current record format
707 0798 3 |
```



```

: 708      0799      FMT_CURR = FMT_ROOT - %FIELDEXPAND(FMT_NEXT,0);
: 709      0800
: 710      0801      ! Start counting conditions from zero
: 711      0802
: 712      0803      COND_NUM = 0;
: 713      0804
: 714      0805      ! Get a line. Check for a header record, and an altseq record
: 715      0806
: 716      0807      GET_LINE();
: 717      0808      IF .LINE_TYPE EQL HEADER_LINE THEN DO_HEADER();
: 718      0809      IF .LINE_TYPE EQL ALTSEQ_LINE THEN DO_ALTSEQ();
: 719      0810
: 720      0811      ! Get the rest of the records
: 721      0812
: 722      0813      WHILE .LINE_TYPE NEQ 0 DO DO_RECORD();
: 723      0814
: 724      0815      ! Exit the loop if this was the last pass
: 725      0816
: 726      0817      IF .PASS EQL 3 THEN EXITLOOP;
: 727      0818
: 728      0819      ! Switch to the temporary file, if needed
: 729      0820
: 730      0821      IF .TMP_FAB[FAB$W_IFI] NEQ 0 THEN CUR_RAB = TMP_RAB[BASE_];
: 731      0822      CUR_RAB[RAB$L_CTX] = SRTTRN$READERR;
: 732      0823      $REQIND( RAB=CUR_RAB[BASE_],-ERR=REPORT_IO_ERROR );
: 733      0824      END;
: 734      0825
: 735      0826
: 736      0827      ! Flush the output, and close the files
: 737      0828
: 738      0829      PUT_LINE();
: 739      0830      CLOSE_FILES();
: 740      0831
: 741      0832      ! Exit with status
: 742      0833
: 743      0834      %IF %BLISS(BLISS16)
: 744      0835      %THEN
: 745      0836          EXST$( .SOR_STS );
: 746      0837      %FI
: 747      0838      RETURN .SOR_STS;
: 748      0839      END;

```

```

: 44 52 4F 43 45 52 3D 53 53 45 43 4F 52 50 2F 0000E P.AAD: .ASCII \!++\
: 53 45 52 44 44 41 3D 53 53 45 43 4F 52 50 2F 00011 P.AAE: .ASCII \!--\
: 55 58 45 44 4E 49 3D 53 53 45 43 4F 52 50 2F 00014 P.AAF: .ASCII \!PROCESS=RECORD\
: 3A 45 43 4E 45 55 51 45 53 28 3D 45 43 4E 45 00023 P.AAG: .ASCII \!PROCESS=TAG\
: 29 43 49 44 43 42 45 0002F P.AAH: .ASCII \!PROCESS=ADDRESS\
: 0003E
: 0003F P.AAI: .ASCII \!PROCESS=INDEX\
: 0004D P.AAJ: .ASCII \!COLLATING_SEQUENCE=(SEQUENCE:EBCDIC)\
: 0005C
: 0006B

```

.PSECT \$PLITS,NOWRT,NOEXE,2

: 0685

: 0778

00

1C
F4
F8
0000V
D8

04
FO

00000052

00000054

00000041

00000049

		31	A8	9F	000B8	PUSHAB	P.AAI		
			0E	DD	000BB	PUSHL	#14		
	69		02	FB	000BD	CALLS	#2, PUT_TEXT		
	08	08	A6	E9	000C0	BLBC	EBCDIC, -8\$	0781	
		3F	A8	9F	000C4	PUSHAB	P.AAJ	0783	
			25	DD	000C7	PUSHL	#37		
	69		02	FB	000C9	CALLS	#2, PUT_TEXT		
0000V	CF		00	FB	000CC	CALLS	#0, DEF_SUBFIELDS	0789	
		E0	A6	DD	000D1	PUSHL	FLD_ROOT	0790	
0000V	CF		01	FB	000D4	CALLS	#1, DEF_FIELDS		
E8	A6	E4	A6	9E	000D9	MOVAB	FMT_ROOT, FMT_CURR	0799	
		FC	A6	D4	000DE	CLRL	COND_NUM	0803	
0000V	CF		00	FB	000E1	CALLS	#0, GET_LINE	0807	
	01	EC	A6	D1	000E6	CMPL	LINE_TYPE, #1	0808	
			05	12	000EA	BNEQ	10\$		
0000V	CF		00	FB	000EC	CALLS	#0, DO_HEADER		
	02	EC	A6	D1	000F1	CMPL	LINE_TYPE, #2	0809	
			05	12	000F5	BNEQ	11\$		
0000V	CF		00	FB	000F7	CALLS	#0, DO_ALTSEQ		
		EC	A6	D5	000FC	TSTL	LINE_TYPE	0813	
			07	13	000FF	BEQL	12\$		
0000V	CF		00	FB	00101	CALLS	#0, DO_RECORD		
			F4	11	00106	BRB	11\$		
	03		66	D1	00108	CMPL	PASS, #3	0817	
			26	13	0010B	BEQL	14\$		
		035A	C7	B5	0010D	TSTW	TMP_FAB+2	0821	
			05	13	00111	BEQL	13\$		
	67	03A8	C7	9E	00113	MOVAB	TMP_RAB, CUR_RAB		
	50		67	D0	00118	MOVL	CUR_RAB, R0	0822	
18	A0	001C10B2	8F	D0	0011B	MOVL	#1839282, 24(R0)		
		0000V	CF	9F	00123	PUSHAB	REPORT_IO_ERROR	0823	
			50	DD	00127	PUSHL	R0		
00000000G	00		02	FB	00129	CALLS	#2, SYS\$REWIND		
		FF	22	31	00130	BRW	1\$	0759	
	6A		00	FB	00133	CALLS	#0, PUT_LINE	0829	
0000V	CF		00	FB	00136	CALLS	#0, CLOSE_FILES	0830	
	50	1C	A6	D0	0013B	MOVL	SOR_STS, R0	0838	
				04	0013F	RET		0839	
				0000	00140	.WORD	Save nothing	0714	
			7E	D4	00142	CLRL	-(SP)		
			5E	DD	00144	PUSHL	SP		
FEA9	7E	04	AC	7D	00146	MOVQ	4(AP), -(SP)		
	CF		03	FB	0014A	CALLS	#3, COND_HAND		
				04	0014F	RET			

; Routine Size: 336 bytes, Routine Base: \$CODE\$ + 005F


```

: 750 0840 1 ROUTINE GETFILENAME (FAB: REF $FAB_DECL) =
: 751 0841 1
: 752 0842 1 ++
: 753 0843 1 FUNCTIONAL DESCRIPTION:
: 754 0844 1
: 755 0845 1 This routine returns a string descriptor for a file.
: 756 0846 1
: 757 0847 1 INPUTS:
: 758 0848 1
: 759 0849 1 FAB Address of FAB
: 760 0850 1
: 761 0851 1 OUTPUTS:
: 762 0852 1
: 763 0853 1 Address of string descriptor for file name
: 764 0854 1
: 765 0855 1 --
: 766 0856 2 BEGIN
: 767 0857 2 LOCAL
: 768 0858 2 NAM: REF $NAM_DECL;
: 769 0859 2
: 770 0860 2 OWN
: 771 0861 2 DESC: VECTOR[2];
: 772 0862 2
: 773 0863 2 NAM = .FAB[FAB$L_NAM];
: 774 0864 2 IF NAM[BASE_] NEQ 0
: 775 0865 2 THEN
: 776 0866 3 BEGIN
: 777 0867 3 IF (DESC[0] = .NAM[NAM$B_RSL]) NEQ 0 THEN DESC[1] = .NAM[NAM$L_RSA]
: 778 0868 3 ELIF (DESC[0] = .NAM[NAM$B_ESL]) NEQ 0 THEN DESC[1] = .NAM[NAM$L_ESA]
: 779 0869 3 ELSE
: 780 0870 4 BEGIN
: 781 0871 4 DESC[0] = .FAB[FAB$B_FNS];
: 782 0872 4 DESC[1] = .FAB[FAB$L_FNA];
: 783 0873 3 END;
: 784 0874 3 END
: 785 0875 2 ELSE
: 786 0876 3 BEGIN
: 787 0877 3 DESC[0] = .FAB[FAB$B_FNS];
: 788 0878 3 DESC[1] = .FAB[FAB$L_FNA];
: 789 0879 2 END;
: 790 0880 2
: 791 0881 2 RETURN DESC[0];
: 792 0882 1 END;
```

.PSECT \$OWN\$,NOEXE,2

00770 DESC: .BLKB 8

.PSECT \$CODE\$,NOWRT,2

0004 00000 GETFILENAME:

52 0000' CF 9E 00002 .WORD Save R2
MOVAB DESC, R2

: 0840
:

SRTRN
V04-000

M 13
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTRN.B32;1

Page 26
(8)

	51	04	AC	D0	00007	MOVL	FAB, R1	:	0863
	50	28	A1	D0	0000B	MOVL	40(R1), NAM	:	
			1A	13	0000F	BEQL	2\$	:	0864
	62	03	A0	9A	00011	MOVZBL	3(NAM), DESC	:	0867
			07	13	00015	BEQL	1\$	:	
04	A2	04	A0	D0	00017	MOVL	4(NAM), DESC+4	:	
			16	11	0001C	BRB	3\$	:	
	62	0B	A0	9A	0001E	MOVZBL	11(NAM), DESC	:	0868
			07	13	00022	BEQL	2\$	:	
04	A2	0C	A0	D0	00024	MOVL	12(NAM), DESC+4	:	
			09	11	00029	BRB	3\$	:	
	62	34	A1	9A	0002B	MOVZBL	52(R1), DESC	:	0877
04	A2	2C	A1	D0	0002F	MOVL	44(R1), DESC+4	:	0878
	50		62	9E	00034	MOVAB	DESC, R0	:	0881
			04	00037	RET			:	0882

; Routine Size: 56 bytes, Routine Base: \$CODE\$ + 01AF


```
: 794 0883 1 ROUTINE REPORT_IO_ERROR (FRAB: REF BLOCK[,BYTE]): CALL =
: 795 0884 1
: 796 0885 1 ++
: 797 0886 1
: 798 0887 1 FUNCTIONAL DESCRIPTION:
: 799 0888 1
: 800 0889 1 This routine signals an I/O error.
: 801 0890 1
: 802 0891 1 INPUTS:
: 803 0892 1
: 804 0893 1 FRAB The FAB or the RAB which got the error
: 805 0894 1 The $L_CTX field of the FAB/RAB must contain the
: 806 0895 1 error code to signal
: 807 0896 1 (SHR$ OPENIN/OPENOUT/READERR/WRITEERR/CLOSEIN/CLOSEOUT)
: 808 0897 1 If FRAB is a RAB, then RAB$L_FAB must point to the FAB
: 809 0898 1 In either case, the FAB must point to a valid NAM block
: 810 0899 1 with both the expanded and resultant name strings in
: 811 0900 1 order for consistent error reporting.
: 812 0901 1
: 813 0902 1 OUTPUTS:
: 814 0903 1
: 815 0904 1 The error is signalled. RMS$_EOF is not signalled
: 816 0905 1
: 817 0906 1 ROUTINE VALUE:
: 818 0907 1
: 819 0908 1 The $L_STS field of FRAB is returned
: 820 0909 1
: 821 0910 1 --
: 822 0911 2 BEGIN
: 823 0912 2 IF .FRAB[RAB$L_STS] NEQ RMS$_EOF
: 824 0913 2 THEN
: 825 0914 2 SIGNAL(.FRAB[FAB$L_CTX], 1,
: 826 0915 3 GETFILENAME((IF .FRAB[FAB$B_BID] EQL FAB$C_BID
: 827 0916 2 THEN FRAB[BASE_] ELSE .FRAB[RAB$L_FAB])),
: 828 0917 2 .FRAB[FAB$L_STS], .FRAB[FAB$L_STV]);
: 829 0918 2
: 830 0919 2 RETURN .FRAB[FAB$L_STS];
: 831 0920 1 END;
```

```
0004 00000 REPORT_IO_ERROR:
                                .WORD Save R2
0001827A 52 04 AC D0 00002      MOVL FRAB, R2
                                8(R2), #98938
                                3$
                                8(R2), -(SP)
                                (R2), #3
                                1$
                                R2
                                2$
                                60(R2)
                                #1, GETFILENAME
                                R0
                                #1
                                0915
                                0914
```



```
B 14
16-Sep-1984 01:11:52      VAX-11 BLISS-32 V4.0-742
14-Sep-1984 13:10:54      [SORT32.SRC]SRTTRN.B32;1
```

Page 28
(9)

00000000G	00	18	A2	DD	00028		PUSHL	24(R2)
	50		05	FB	00028		CALLS	#5, LIB\$SIGNAL
		08	A2	D0	00032	3\$:	MOVL	8(R2), R0
				04	00036		RET	

0919
0920

```
; Routine Size: 55 bytes,    Routine Base: $CODE$ + 01E7
```



```

: 833      0921 1 ROUTINE GET_QUAL(
: 834      0922 1     DESC: REF VECTOR[2]
: 835      0923 1     ): NOVALUE =
: 836      0924 2     BEGIN
: 837      0925 2     MACRO
: 838      M 0926 2     IS_QUAL(X) = (IF .D[0] LSS %CHARCOUNT(X) THEN FALSE ELSE
: 839      M 0927 2     CH$EQL(%CHARCOUNT(X), UPLIT BYTE(X), %CHARCOUNT(X), .D[1])) %,
: 840      M 0928 2     LOCC(X) = (LOCAL T; T = CH$FIND CH(.D[0], .D[1], X);
: 841      0929 2     IF CH$FAIL(.T) THEN T = CH$PLUS(.D[1], .D[0]); .T) %;
: 842      0930 2     LOCAL
: 843      0931 2     D; REF VECTOR[2],
: 844      0932 2     F R;
: 845      0933 2
: 846      0934 2     ! G a local pointer to the descriptor
: 847      0935 2     !
: 848      0936 2     D = SC[0];
: 849      0937 2
: 850      0938 2     IF QUAL('/S32') THEN SWITCH_S32 = TRUE
: 851      0939 2     ELSE IF QUAL('/S11') THEN SWITCH_S32 = FALSE
: 852      0940 2     ELSE
: 853      0941 2     RE R_ERROR(SRTTRN$_INV_QUAL);
: 854      0942 2
: 855      0943 2     ! Advance past this qualifier
: 856      0944 2     !
: 857      0945 2     IF (D[0] = .D[0] - 1) EQL 0 THEN RETURN;
: 858      0946 2     D[1] = .D[1] + 1;
: 859      0947 2     PTR = MINA( LOCC(%C'/'), LOCC(%C' ') );
: 860      0948 2     D[0] = .D[0] - CH$DIFF(.PTR, .D[1]);
: 861      0949 2     D[1] = .PTR;
: 862      0950 1     END;
```

.PSECT \$SPLITS\$,NOWRT,NOEXE,2

```

32 33 53 2F 00072 P.AAK: .ASCII \S32\
31 31 53 2F 00076 P.AAL: .ASCII \S11\
```

.PSECT \$CODES\$,NOWRT,2

```

                                001C 00000 GET_QUAL:
                                .WORD Save R2,R3,R4
                                52      04      AC D0 00002      MOVL  DESC, D
                                50      D4 00006      CLRL  R0
                                04      62 D1 00008      CMPL  (D), #4
                                04      18 0000B      BGEQ  1$
                                50      D6 0000D      INCL  R0
                                0F      11 0000F      BRB   2$
                                04      B2 0000' CF D1 00011 1$: CMPL  P.AAK, @4(D)
                                0000' CF 07 12 00017      BNEQ  2$
                                01      D0 00019      MOVL  #1, SWITCH_S32
                                1D      11 0001E      BRB   4$
                                04      0E 0000' 50 E8 00020 2$: BLBS  R0, 3$
                                04      B2 0000' CF D1 00023      CMPL  P.AAL, @4(D)
                                06      12 00029      BNEQ  3$
```

```

: 0921
: 0936
: 0938
:
:
:
:
:
:
: 0939
:
```


		0000'	CF	D4	0002B	CLRL	SWITCH_S32	:	
			0C	11	0002F	BRB	4\$	:	
	FDA6	CF	8F	DD	00031	PUSHL	#1867836	:	0941
			01	FB	00037	CALLS	#1, SOR_ERROR	:	
				04	0003C	RET		:	
			62	D7	0003D	DECL	(D)	:	0945
			3F	13	0003F	BEQL	10\$	:	
		04	A2	D6	00041	INCL	4(D)	:	0946
63		53	A2	D0	00044	MOVL	4(D), R3	:	0947
		62	2F	3A	00048	LOCC	#47, (D), (R3)	:	
			02	12	0004C	BNEQ	5\$	:	
			51	D4	0004E	CLRL	R1	:	
		54	51	D0	00050	MOVL	R1, T	:	
			04	12	00053	BNEQ	6\$	:	
54		53	62	C1	00055	ADDL3	(D), R3, T	:	
63		62	20	3A	00059	LOCC	#32, (D), (R3)	:	
			02	12	0005D	BNEQ	7\$	:	
			51	D4	0005F	CLRL	R1	:	
		50	51	D0	00061	MOVL	R1, T	:	
			04	12	00064	BNEQ	8\$	:	
50		53	62	C1	00066	ADDL3	(D), R3, T	:	
		51	54	D0	0006A	MOVL	T, R1	:	
		50	51	D1	0006D	CMPL	R1, T	:	
			03	1B	00070	BLEQU	9\$	:	
		51	50	D0	00072	MOVL	T, R1	:	
50		53	51	C3	00075	SUBL3	PTR, R3, R0	:	0948
		62	50	C0	00079	ADDL2	R0, (D)	:	0949
	04	A2	51	D0	0007C	MOVL	PTR, 4(D)	:	0950
			04	00080	10\$:	RET		:	

; Routine Size: 129 bytes, Routine Base: \$CODE\$ + 021E


```
: 864 0951 1 ROUTINE GET_FILE_SPEC(
: 865 0952 1   DESC: REF VECTOR[2],
: 866 0953 1   FNA: REF VECTOR[1],
: 867 0954 1   FNS: REF VECTOR[1, BYTE]
: 868 0955 1   ): NOVALUE =
: 869 0956 1   ++
: 870 0957 1
: 871 0958 1   FUNCTIONAL DESCRIPTION:
: 872 0959 1
: 873 0960 1       Extract one file name string from DESC, storing the address and length
: 874 0961 1       in FNA and FNS respectively, and updating DESC.
: 875 0962 1
: 876 0963 1   FORMAL PARAMETERS:
: 877 0964 1
: 878 0965 1       NONE
: 879 0966 1
: 880 0967 1   IMPLICIT INPUTS:
: 881 0968 1
: 882 0969 1       NONE
: 883 0970 1
: 884 0971 1   IMPLICIT OUTPUTS:
: 885 0972 1
: 886 0973 1       NONE
: 887 0974 1
: 888 0975 1   ROUTINE VALUE:
: 889 0976 1
: 890 0977 1       NONE
: 891 0978 1
: 892 0979 1   SIDE EFFECTS:
: 893 0980 1
: 894 0981 1       NONE
: 895 0982 1
: 896 0983 1   --
: 897 0984 2   BEGIN
: 898 0985 2   LOCAL
: 899 0986 2   PTR;
: 900 0987 2
: 901 0988 2   ! Find the beginning of the file name string
: 902 0989 2   !
: 903 0990 2   WHILE TRUE DO
: 904 0991 2   BEGIN
: 905 0992 2   PTR = CH$FIND NOT CH(.DESC[0], .DESC[1], %C' ');
: 906 0993 2   IF CH$FAIL(.PTR) THEN PTR = CH$PLUS(.DESC[1], .DESC[0]);
: 907 0994 2   DESC[0] = CH$DIFF(CH$PLUS(.DESC[1], .DESC[0]), .PTR);
: 908 0995 2   DESC[1] = .PTR;
: 909 0996 2   IF .DESC[0] EQL 0 THEN EXITLOOP;
: 910 0997 2   IF CH$RCHAR(.DESC[1]) NEQ %C'/' THEN EXITLOOP;
: 911 0998 2   GET_QUAL(DESC[0]);
: 912 0999 2   END;
: 913 1000 2
: 914 1001 2   ! Store the file name string's address
: 915 1002 2   !
: 916 1003 2   FNA[0] = .PTR;
: 917 1004 2
: 918 1005 2   ! Check for an equal sign
: 919 1006 2   !
: 920 1007 2   IF CH$RCHAR(.PTR) EQL %C'=';
```

```

: 921      1008      2
: 922      1009
: 923      1010
: 924      1011
: 925      1012
: 926      1013
: 927      1014
: 928      1015
: 929      1016
: 930      1017
: 931      1018
: 932      1019
: 933      1020
: 934      1021
: 935      1022
: 936      1023
: 937      1024
: 938      1025
: 939      1026
: 940      1027
: 941      1028
: 942      1029
: 943      1030
: 944      1031
: 945      1032
: 946      1033
: 947      1034
: 948      1035
: 949      1036
: 950      1037
: 951      1038
: 952      1039
: 953      1040
: 954      1041
: 955      1042      1

THEN
BEGIN
DESC[0] = .DESC[0] - 1;
DESC[1] = .DESC[1] + 1;
FNS[0] = CH$DIFF(.DESC[1], .FNA[0]);
RETURN;
END;

! Find the next space (or equal sign) that is not enclosed in quotes.
!
WHILE TRUE DO
BEGIN
LOCAL
QUO;
PTR = CH$FIND CH(.DESC[0], .DESC[1], %C' ');
IF CH$FAIL(.PTR) THEN PTR = CH$PLUS(.DESC[1], .DESC[0]);

QUO = CH$FIND CH(CH$DIFF(.PTR, .DESC[1]), .DESC[1], %C'=');
IF NOT CH$FAIL(.QUO) THEN IF .QUO LSSA .PTR THEN PTR = .QUO;

QUO = CH$FIND CH(CH$DIFF(.PTR, .DESC[1]), .DESC[1], %C'');
DESC[0] = CH$DIFF(CH$PLUS(.DESC[1], .DESC[0]), .PTR);
DESC[1] = .PTR;
IF CH$FAIL(.QUO) THEN EXITLOOP;
PTR = CH$FIND CH(.DESC[0], .DESC[1], %C'');
IF CH$FAIL(.PTR) THEN EXITLOOP;
DESC[0] = CH$DIFF(CH$PLUS(.DESC[1], .DESC[0]), .PTR);
DESC[1] = .PTR;
END;

! Store the file name string's length
!
FNS[0] = CH$DIFF(.DESC[1], .FNA[0]);
END;
```

003C 00000 GET_FILE_SPEC:							
		53	04	AC	D0 00002	Save R2,R3,R4,R5	: 0951
		52	04	A3	9E 00006	DESC, R3	: 0992
00	B2	63		20	3B 0000A 1\$:	4(R3), R2	
				02	12 0000F	#32, (R3), a0(R2)	
				51	D4 00011	2\$	
		54		51	D0 00013 2\$:	R1	
				04	12 00016	R1, PTR	
				63	C1 00018	3\$	: 0993
	54	62		63	C1 0001C 3\$:	(R3), (R2), PTR	
	50	62		54	C3 00020	(R3), (R2), R0	: 0994
	63	50		54	D0 00024	PTR, R0, (R3)	
		62		63	D5 00027	PTR, (R2)	: 0995
				0F	13 00029	(R3)	: 0996
		2F	00	B2	91 0002B	4\$	
				09	12 0002F	a0(R2), #47	: 0997
						4\$	

				53	DD	00031	PUSHL	R3		0998
		FF47	CF	01	FB	00033	CALLS	#1, GET_QUAL		
				D0	11	00038	BRB	1\$		0990
		08	BC	54	D0	0003A	4\$:	MOVL	PTR, @FNA	1003
			3D	64	91	0003E	CMPB	(PTR), #61		1007
				06	12	00041	BNEQ	5\$		
				63	D7	00043	DECL	(R3)		1010
				62	D6	00045	INCL	(R2)		1011
				66	11	00047	BRB	12\$		1012
00	B2			20	3A	00049	5\$:	LOCC	#32, (R3), @0(R2)	1022
				02	12	0004E	BNEQ	6\$		
				51	D4	00050	CLRL	R1		
			54	51	D0	00052	6\$:	MOVL	R1, PTR	
				04	12	00055	BNEQ	7\$		1023
	54		62	63	C1	00057	ADDL3	(R3), (R2), PTR		
	51		54	62	C3	0005B	7\$:	SUBL3	(R2), PTR, R1	1025
00	B2		51	3D	3A	0005F	LOCC	#61, R1, @0(R2)		
				02	12	00064	BNEQ	8\$		
				51	D4	00066	CLRL	R1		
			55	51	D0	00068	8\$:	MOVL	R1, QUO	
				08	13	0006B	BEQL	9\$		1026
			54	55	D1	0006D	CMPL	QUO, PTR		
				03	1E	00070	BGEQU	9\$		
			54	55	D0	00072	MOVL	QUO, PTR		
	51		54	62	C3	00075	9\$:	SUBL3	(R2), PTR, R1	1028
00	B2		51	22	3A	00079	LOCC	#34, R1, @0(R2)		
				02	12	0007E	BNEQ	10\$		
				51	D4	00080	CLRL	R1		
			55	51	D0	00082	10\$:	MOVL	R1, QUO	
	50		62	63	C1	00085	ADDL3	(R3), (R2), R0		1029
	63		50	54	C3	00089	SUBL3	PTR, R0, (R3)		
			62	54	D0	0008D	MOVL	PTR, (R2)		1030
				55	D5	00090	TSTL	QUO		1031
				1B	13	00092	BEQL	12\$		
00	B2		63	22	3A	00094	LOCC	#34, (R3), @0(R2)		1032
				02	12	00099	BNEQ	11\$		
				51	D4	0009B	CLRL	R1		
			54	51	D0	0009D	11\$:	MOVL	R1, PTR	
				0D	13	000A0	BEQL	12\$		1033
	50		62	63	C1	000A2	ADDL3	(R3), (R2), R0		1034
	63		50	54	C3	000A6	SUBL3	PTR, R0, (R3)		
			62	54	D0	000AA	MOVL	PTR, (R2)		1035
				9A	11	000AD	BRB	5\$		1018
0C	BC		62	08	BC	83	12\$:	SUBB3	@FNA, (R2), @FNS	1040
					04	000B5	RET			1042

; Routine Size: 182 bytes, Routine Base: \$CODE\$ + 029F

```

: 957      1043 1 ROUTINE GET FOREIGN(
: 958      1044 1     DESC: REF VECTOR[2]
: 959      1045 1     ): NOVALUE =
: 960      1046 2     BEGIN
: 961      1047 2     %IF %BLISS(BLISS32) %THEN
: 962      1048 2     MACRO
: 963      1049 2     PROMPT = 'SRTRN> ' %;
: 964      1050 2     %ELSE
: 965      1051 2     MACRO
: 966      1052 2     PROMPT = 'TRN> ' %;
: 967      1053 2     %FI
: 968      1054 2     LOCAL
: 969      1055 2     STATUS;
: 970      1056 2
: 971      1057 2
: 972      1058 2 %IF %BLISS(BLISS32) !AND K_NATIVE
: 973      1059 2 %THEN
: 974      1060 2     BEGIN
: 975      1061 2     EXTERNAL ROUTINE LIB$GET FOREIGN;
: 976      1062 2     STATUS = LIB$GET FOREIGN(DESC[0]);
: 977      1063 2     IF NOT .STATUS THEN SIGNAL(.STATUS);
: 978      1064 2     END
: 979      1065 2 %ELSE
: 980      1066 2     BEGIN
: 981      1067 2     LITERAL
: 982      1068 2     K_VICS = FALSE;
: 983      1069 2
: 984      1070 2     %IF K_VICS
: 985      1071 2     %THEN
: 986      1072 2     EXTERNAL ROUTINE $TRNGC;
: 987      1073 2     STATUS = $TRNGC(DESC[0]);
: 988      1074 2     %ELSE
: 989      1075 2     LIBRARY 'SYS$LIBRARY:FCS11';
: 990      1076 2     FSR$Z$(1);
: 991      1077 2     GCMLB$( GCMLB, 1, 'trn' );
: 992      1078 2     EXTERNAL $DSW;
: 993      1079 2     LOCAL STATUS;
: 994      1080 2     ALUN$$ ( LUN_K MSG, 'TI', 0 );
: 995      1081 2     IF .SDSW NEQ IS_SUC THEN RETURN;
: 996      1082 2     DO
: 997      1083 2     BEGIN
: 998      1084 2     GCML$( GCMLB, UPLIT BYTE(%X'0A',PROMPT), %CHARCOUNT(PROMPT) );
: 999      1085 2     STATUS = .GCMLB[G_ERR];
: 1000     1086 2     DESC[0] = .GCMLB[G_CMLD_L];
: 1001     1087 2     DESC[1] = .GCMLB[G_CMLD_A];
: 1002     1088 2     IF .STATUS EQL GE_EOF
: 1003     1089 2     THEN
: 1004     1090 2     EXST$( EX$SUC );
: 1005     1091 2     END
: 1006     1092 2     WHILE .DESC[ 0 ] EQL 0;
: 1007     1093 2     %FI
: 1008     1094 2
: 1009     1095 2     IF .STATUS NEQ 0 THEN SIGNAL(SRTRN$_NO_COMMAND,0,.STATUS);
: 1010     1096 2     END
: 1011     1097 2     %FI;
: 1012     1098 2     END;
```



```

16-Sep-1984 01:11:52      VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:54      [SORT32.SRC]SRTTRN.B32;1

```

Page 35
(12)

```
0000 00000 GET_FOREIGN:
```

04

AC	DD	00002
01	FB	00005
50	E8	0000C
50	DD	0000F
01	FB	00011
	04	00018

```

        .WORD      Save nothing
        PUSHL      DESC
        CALLS      #1, LIB$GET_FOREIGN
        BLBS       STATUS, 1$
        PUSHL      STATUS
        CALLS      #1, LIB$SIGNAL
        RET

```

: 1043
 : 1062
 :
 : 1063
 :
 :
 :
 : 1098

00000000G 00

```
; Routine Size: 25 bytes,    Routine Base: $CODE$ + 0355
```

**SRT
V04**

[illegible]

```
1014 1099 1 ROUTINE INIT_IO: NOVALUE =
1015 1100 1
1016 1101 1 ++
1017 1102 1
1018 1103 1 FUNCTIONAL DESCRIPTION:
1019 1104 1
1020 1105 1 Initialization routine.
1021 1106 1 Translates foreign command line, opens input and output files.
1022 1107 1
1023 1108 1 FORMAL PARAMETERS:
1024 1109 1
1025 1110 1 NONE
1026 1111 1
1027 1112 1 IMPLICIT INPUTS:
1028 1113 1
1029 1114 1 NONE
1030 1115 1
1031 1116 1 IMPLICIT OUTPUTS:
1032 1117 1
1033 1118 1 NONE
1034 1119 1
1035 1120 1 ROUTINE VALUE:
1036 1121 1
1037 1122 1 NONE
1038 1123 1
1039 1124 1 SIDE EFFECTS:
1040 1125 1
1041 1126 1 NONE
1042 1127 1
1043 1128 1 --
1044 1129 2 BEGIN
1045 1130 2 %IF %BLISS(BLISS32) %THEN LOCAL %ELSE OWN %FI
1046 1131 2 COM_BUF: VECTOR[256,BYTE]; ! Holds foreign command line
1047 1132 2 LOCAL
1048 1133 2 OPT_DESC: VECTOR[2], ! Command line desc
1049 1134 2 DUMMY_L,
1050 1135 2 DUMMY_B: BYTE,
1051 1136 2 STATUS;
1052 1137 2
1053 1138 2 OPT_DESC[0] = %ALLOCATION(COM_BUF);
1054 1139 2 OPT_DESC[1] = COM_BUF[0];
1055 1140 2
1056 1141 2 ! Get command line options if any
1057 1142 2 !
1058 1143 2 GET_FOREIGN(OPT_DESC[0]);
1059 1144 2
1060 1145 2 %IF %BLISS(BLISS32) AND K_NATIVE
1061 1146 2 %THEN
1062 1147 2 BEGIN
1063 1148 2
1064 1149 2 SRTTRN inp-file out-file
1065 1150 2 SRTTRN inp-file
1066 1151 2 SRTTRN
1067 1152 2
1068 1153 2 GET_FILE_SPEC(OPT_DESC, INP_FAB[FAB$L_FNA], INP_FAB[FAB$B_FNS]);
1069 1154 2 GET_FILE_SPEC(OPT_DESC, OUT_FAB[FAB$L_FNA], OUT_FAB[FAB$B_FNS]);
1070 1155 2 GET_FILE_SPEC(OPT_DESC, DUMMY_L, DUMMY_B);
```



```
IF .DUMMY_B NEQ 0 THEN SOR_ERROR(SRTTRN$_INV_QUAL);
END
%ELSE
BEGIN
    SRTTRN out-file = inp-file
    SRTTRN = inp-file
    GET FILE SPEC(OPT_DESC, OUT_FAB[FAB$L_FNA], OUT_FAB[FAB$B_FNS]);
    IF %HSEQ[.OUT_FAB[FAB$B_FNS], .OUT_FAB[FAB$L_FNA], LENADR_('=')]
    THEN
        OUT_FAB[FAB$B_FNS] = 0
    ELSE
        BEGIN
            GET FILE SPEC(OPT_DESC, DUMMY_L, DUMMY_B);
            IF %HSEQ(.DUMMY_B, .DUMMY_L, LENADR_('='))
            THEN
                SOR_ERROR(SRTTRN$_INV_QUAL);
            END;
            GET FILE SPEC(OPT_DESC, INP_FAB[FAB$L_FNA], INP_FAB[FAB$B_FNS]);
            GET FILE SPEC(OPT_DESC, DUMMY_L, DUMMY_B);
            IF .DUMMY_B NEQ 0 THEN SOR_ERROR(SRTTRN$_INV_QUAL);
        END
    %FI;
    ! Initialize RMS
    !
    %IF %BLISS(BLISS16)
    %THEN
        %RMS_INITIF();
        INP_FAB[FAB$B_LCH] = LUN_K_INP;
        OUT_FAB[FAB$B_LCH] = LUN_K_OUT;
        TMP_FAB[FAB$B_LCH] = LUN_K_TMP;
    %FI
    ! Default the file names as needed
    !
    IF .INP_FAB[FAB$B_FNS] EQL 0
    THEN
        BEGIN
            INP_FAB[FAB$L_FNA] = UPLIT BYTE('SYSS$INPUT');
            INP_FAB[FAB$B_FNS] = %CHARCOUNT('SYSS$INPUT');
            IF .OUT_FAB[FAB$B_FNS] EQL 0
            THEN
                BEGIN
                    OUT_FAB[FAB$L_FNA] = UPLIT BYTE('SYSS$OUTPUT');
                    OUT_FAB[FAB$B_FNS] = %CHARCOUNT('SYSS$OUTPUT');
                END;
            END;
        END;
    ! There is no default extension for /S32
    !
    IF .SWITCH_S32
    THEN
        BEGIN
            SOR_ERROR(SRTTRN$_LINE_NUM);    ! ???
            INP_FAB[FAB$L_DNA] = 0;
```

```
: 1128      1213      3      INP_FAB[FAB$B_DNS] = 0;
: 1129      1214      2      END;
: 1130      1215      2
: 1131      1216      2
: 1132      1217      2      ! Open input file
: 1133      1218      2      INP_FAB[FAB$L_CTX] = SRTTRN$_OPENIN;
: 1134      1219      2      INP_RAB[RAB$L_CTX] = SRTTRN$_OPENIN;
: 1135      1220      2      $OPEN( FAB=INP_FAB[BASE_], ERR=REPORT_IO_ERROR );
: 1136      1221      2      $CONNECT( RAB=INP_RAB[BASE_], ERR=REPORT_IO_ERROR );
: 1137      1222      2      CUR_RAB = INP_RAB[BASE_];
: 1138      1223      2
: 1139      1224      2      ! If the input file is not on a local file-oriented device,
: 1140      1225      2      ! we want to make a copy of the file.
: 1141      1226      2
: 1142      1227      2      IF
: 1143      1228      2      %IF %BLISS(BLISS32)
: 1144      1229      2      %THEN
: 1145      1230      2      NOT .BLOCK[INP_FAB[FAB$L_DEV], DEV$V_FOD] OR
: 1146      1231      2      .BLOCK[INP_FAB[FAB$L_DEV], DEV$V_NET]
: 1147      1232      2      %ELSE
: 1148      1233      2      NOT .INP_FAB[FAB$V_MDI] AND      ! Not MULTI-DIRECTORY
: 1149      1234      2      NOT .INP_FAB[FAB$V_SDI] AND      ! Not SINGLE DIRECTORY
: 1150      1235      2      NOT .INP_FAB[FAB$V_SQD]      ! Not SEQUENTIAL DEVICE
: 1151      1236      2      %FI
: 1152      1237      2      THEN
: 1153      1238      3      BEGIN
: 1154      1239      3      TMP_FAB[FAB$L_CTX] = SRTTRN$_OPENOUT;
: 1155      1240      3      TMP_RAB[RAB$L_CTX] = SRTTRN$_OPENOUT;
: 1156      1241      3      $CREATE( FAB=TMP_FAB[BASE_], ERR=REPORT_IO_ERROR );
: 1157      1242      3      $CONNECT( RAB=TMP_RAB[BASE_], ERR=REPORT_IO_ERROR );
: 1158      1243      2      END;
: 1159      1244      2
: 1160      1245      2      ! Open output file
: 1161      1246      2
: 1162      1247      2      OUT_FAB[FAB$L_CTX] = SRTTRN$_OPENOUT;
: 1163      1248      2      OUT_RAB[RAB$L_CTX] = SRTTRN$_OPENOUT;
: 1164      1249      2      $CREATE( FAB=OUT_FAB[BASE_], ERR=REPORT_IO_ERROR );
: 1165      1250      2      $CONNECT( RAB=OUT_RAB[BASE_], ERR=REPORT_IO_ERROR );
: 1166      1251      2
: 1167      1252      2      ! Issue lines indicating the translation
: 1168      1253      2
: 1169      1254      3      BEGIN
: 1170      1255      3      LOCAL Z: REF VECTOR[2];
: 1171      1256      3      PUT_TEXT(LENADR_('!++'));
: 1172      1257      3      PUT_LINE();
: 1173      1258      3      PUT_TEXT(LENADR_('! SRTTRN'));
: 1174      1259      3      PUT_LINE();
: 1175      1260      3      Z = GETFILENAME(INP_FAB[BASE_]);
: 1176      1261      3      PUT_TEXT(LENADR_('!T,%CHAR(9T),.Z[0],.Z[1]));
: 1177      1262      3      PUT_LINE();
: 1178      1263      3      Z = GETFILENAME(OUT_FAB[BASE_]);
: 1179      1264      3      PUT_TEXT(LENADR_('!T,%CHAR(9T),.Z[0],.Z[1]));
: 1180      1265      3      PUT_LINE();
: 1181      1266      2      END;
: 1182      1267      2
: 1183      1268      1      END;
```



```
.PSECT $SPLITS,NOWRT,NOEXE,2

54 54 55 50 4E 49 24 53 59 53 0007A P.AAM: .ASCII \SYSS$INPUT\
55 55 50 54 55 4F 24 53 59 53 00083 P.AAN: .ASCII \SYSS$OUTPUT\
      4E 52 54 54 52 53 2B 2B 21 0008D P.AAO: .ASCII \!++\
      20 21 00090 P.AAP: .ASCII \! SRTTRN\
      09 21 00098 P.AAQ: .ASCII \!<9>
      09 21 0009A P.AAR: .ASCII \!<9>

.EXTRN SYSS$OPEN, SYSS$CONNECT
.EXTRN SYSS$CREATE

.PSECT $CODE$,NOWRT,2

      03FC 00000 INIT_IO: .WORD Save R2,R3,R4,R5,R6,R7,R8,R9
59 0000V CF 9E 00002 MOVAB PUT_LINE, R9
58 0000V CF 9E 00007 MOVAB PUT_TEXT, R8
57 00000000G 00 9E 0000C MOVAB SYSS$CREATE, R7
56 00000000G 00 9E 00013 MOVAB SYSS$CONNECT, R6
55 0000' CF 9E 0001A MOVAB P.AAM, R5
54 FE56 CF 9E 0001F MOVAB REPORT IO ERROR, R4
53 0000' CF 9E 00024 MOVAB INP_FAB+52, R3
5E FEFO CE 9E 00029 MOVAB -272(SP), SP
08 AE 0100 8F 3C 0002E MOVZWL #256, OPT_DESC
0C AE 10 AE 9E 00034 MOVAB COM_BUF, OPT_DESC+4
      08 AE 9F 00039 PUSHAB OPT_DESC
016E C4 01 FB 0003C CALLS #1, GET_FOREIGN
      53 DD 00041 PUSHL R3
      F8 A3 9F 00043 PUSHAB INP_FAB+44
      10 AE 9F 00046 PUSHAB OPT_DESC
00B8 C4 03 FB 00049 CALLS #3, GET_FILE_SPEC
      03E8 C3 9F 0004E PUSHAB OUT_FAB+52
      03E0 C3 9F 00052 PUSHAB OUT_FAB+44
      10 AE 9F 00056 PUSHAB OPT_DESC
00B8 C4 03 FB 00059 CALLS #3, GET_FILE_SPEC
      5E DD 0005E PUSHL SP
      08 AE 9F 00060 PUSHAB DUMMY_L
      10 AE 9F 00063 PUSHAB OPT_DESC
00B8 C4 03 FB 00066 CALLS #3, GET_FILE_SPEC
      6E 95 0006B TSTB DUMMY_B
      0B 13 0006D BEQL 1$
FE19 C4 001C803C 8F DD 0006F PUSHL #1867836
      01 FB 00075 CALLS #1, SOR_ERROR
      63 95 0007A 1$: TSTB INP_FAB+52
      18 12 0007C BNEQ 2$
      F8 A3 65 9E 0007E MOVAB P.AAM, INP_FAB+44
      63 09 90 00082 MOVAB #9, INP_FAB+52
      03E8 C3 95 00085 TSTB OUT_FAB+52
      0B 12 00089 BNEQ 2$
03E0 C3 09 A5 9E 0008B MOVAB P.AAN, OUT_FAB+44
03E8 C3 0A 90 00091 MOVAB #10, OUT_FAB+52
      11 0000' CF E9 00096 2$: BLBC SWITCH S32, 3$
FE19 C4 001C8058 8F DD 0009B PUSHL #1867864
      FC A3 D4 000A6 CALLS #1, SOR_ERROR
      CLRL INP_FAB+48
```

			01	A3	94	000A9	CLRB	INP_FAB+53	1213	
	E4	A3	001C109C	8F	DO	000AC	3\$:	MOVL	#1839260, INP_FAB+24	1218
	34	A3	001C109C	8F	DO	000B4		MOVL	#1839260, INP_RAB+24	1219
				54	DD	000BC		PUSHL	R4	1220
			CC	A3	9F	000BE		PUSHAB	INP_FAB	
	00000000G	00		02	FB	000C1		CALLS	#2, -SYSS\$OPEN	
				54	DD	000C8		PUSHL	R4	1221
			1C	A3	9F	000CA		PUSHAB	INP_RAB	
		66		02	FB	000CD		CALLS	#2, -SYSS\$CONNECT	
	FE68	C3	1C	A3	9E	000D0		MOVAB	INP_RAB, CUR_RAB	1222
	OD	A3		06	E1	000D6		BBC	#6, -INP_FAB+65, 4\$	1230
05	OD	A3		05	E1	000DB		BBC	#5, -INP_FAB+65, 5\$	1231
24	01D8	C3	001C10A4	8F	DO	000E0	4\$:	MOVL	#1839268, TMP_FAB+24	1239
	0228	C3	001C10A4	8F	DO	000E9		MOVL	#1839268, TMP_RAB+24	1240
				54	DD	000F2		PUSHL	R4	1241
			01C0	C3	9F	000F4		PUSHAB	TMP_FAB	
		67		02	FB	000F8		CALLS	#2, -SYSS\$CREATE	
				54	DD	000FB		PUSHL	R4	1242
			0210	C3	9F	000FD		PUSHAB	TMP_RAB	
		66		02	FB	00101		CALLS	#2, -SYSS\$CONNECT	
	03CC	C3	001C10A4	8F	DO	00104	5\$:	MOVL	#1839268, OUT_FAB+24	1247
	041C	C3	001C10A4	8F	DO	0010D		MOVL	#1839268, OUT_RAB+24	1248
				54	DD	00116		PUSHL	R4	1249
			03B4	C3	9F	00118		PUSHAB	OUT_FAB	
		67		02	FB	0011C		CALLS	#2, -SYSS\$CREATE	
				54	DD	0011F		PUSHL	R4	1250
			0404	C3	9F	00121		PUSHAB	OUT_RAB	
		66		02	FB	00125		CALLS	#2, -SYSS\$CONNECT	
			13	A5	9F	00128		PUSHAB	P.AAO	1256
				03	DD	0012B		PUSHL	#3	
		68		02	FB	0012D		CALLS	#2, PUT_TEXT	
		69		00	FB	00130		CALLS	#0, PUT_LINE	1257
			16	A5	9F	00133		PUSHAB	P.AAP	1258
				08	DD	00136		PUSHL	#8	
		68		02	FB	00138		CALLS	#2, PUT_TEXT	
		69		00	FB	0013B		CALLS	#0, PUT_LINE	1259
			CC	A3	9F	0013E		PUSHAB	INP_FAB	1260
	C8	A4		01	FB	00141		CALLS	#1, -GETFILENAME	
		52		50	DO	00145		MOVL	R0, Z	
		7E		62	7D	00148		MOVQ	(Z), -(SP)	1261
			1E	A5	9F	0014B		PUSHAB	P.AAQ	
				02	DD	0014E		PUSHL	#2	
		68		04	FB	00150		CALLS	#4, PUT_TEXT	
		69		00	FB	00153		CALLS	#0, PUT_LINE	1262
			03B4	C3	9F	00156		PUSHAB	OUT_FAB	1263
				01	FB	0015A		CALLS	#1, -GETFILENAME	
	C8	A4		50	DO	0015E		MOVL	R0, Z	
		52		62	7D	00161		MOVQ	(Z), -(SP)	1264
		7E		A5	9F	00164		PUSHAB	P.AAR	
			20	02	DD	00167		PUSHL	#2	
		68		04	FB	00169		CALLS	#4, PUT_TEXT	
		69		00	FB	0016C		CALLS	#0, PUT_LINE	1265
				04	0016F			RET		1268

; Routine Size: 368 bytes, Routine Base: \$CODE\$ + 036E


```
: 1185 1269 1 ROUTINE DO_HEADER: NOVALUE =
: 1186 1270 1
: 1187 1271 1 ++
: 1188 1272 1
: 1189 1273 1 FUNCTIONAL DESCRIPTION:
: 1190 1274 1
: 1191 1275 1     Get the header record information.
: 1192 1276 1     This includes /process, /nostrip, /collating sequence,
: 1193 1277 1     and normal sequence (ascending or descending) info.
: 1194 1278 1
: 1195 1279 1 FORMAL PARAMETERS:
: 1196 1280 1
: 1197 1281 1     NONE
: 1198 1282 1
: 1199 1283 1 IMPLICIT INPUTS:
: 1200 1284 1
: 1201 1285 1     NONE
: 1202 1286 1
: 1203 1287 1 IMPLICIT OUTPUTS:
: 1204 1288 1
: 1205 1289 1     NONE
: 1206 1290 1
: 1207 1291 1 ROUTINE VALUE:
: 1208 1292 1
: 1209 1293 1     Status code.
: 1210 1294 1
: 1211 1295 1 SIDE EFFECTS:
: 1212 1296 1
: 1213 1297 1     NONE
: 1214 1298 1
: 1215 1299 1 --
: 1216 1300 2 BEGIN
: 1217 1301 2
: 1218 1302 2 IF .PASS EQL 1
: 1219 1303 2 THEN
: 1220 1304 2 BEGIN
: 1221 1305 2 LOCAL
: 1222 1306 2 PTR;
: 1223 1307 2
: 1224 1308 2     ! Convert to upper case
: 1225 1309 2
: 1226 1310 2     UPCASE(.INP_REC_SIZ, INP_BUF);
: 1227 1311 2
: 1228 1312 2     ! Determine if 'normal' is ascending or descending
: 1229 1313 2
: 1230 1314 2     SELECTONE .INP_BUF[17] OF
: 1231 1315 2         SET
: 1232 1316 2         ['A']:          ASCENDING = TRUE;
: 1233 1317 2         ['D']:          ASCENDING = FALSE;
: 1234 1318 2         [OTHERWISE]:    SOR_ERROR(SRTRN$_INV_SEQ);
: 1235 1319 2         TES;
: 1236 1320 2
: 1237 1321 2     ! Get key stripping information
: 1238 1322 2
: 1239 1323 2     SELECTONE .INP_BUF[27] OF
: 1240 1324 2         SET
: 1241 1325 2         [' ']:          NOSTRIP = TRUE;
```


1242	1326	3
1243	1327	3
1244	1328	3
1245	1329	3
1246	1330	3
1247	1331	3
1248	1332	3
1249	1333	3
1250	1334	3
1251	1335	3
1252	1336	3
1253	1337	3
1254	1338	3
1255	1339	3
1256	1340	3
1257	1341	3
1258	1342	3
1259	1343	3
1260	1344	3
1261	1345	3
1262	1346	3
1263	1347	3
1264	1348	3
1265	1349	3
1266	1350	3
1267	1351	2
1268	1352	2
1269	1353	2
1270	1354	2
1271	1355	1

```

['X']:          NOSTRIP = FALSE;
[OTHERWISE]:    SOR_ERROR(SRTTRN$_INV_STRIP);
TES;

IF CH$NEQ(LENADR_('SORT'), 4, INP_BUF[6], %C' ')
THEN
    PROCESS = %C' '
ELSE
    PROCESS = CH$RCHAR(INP_BUF[6+4]);

SELECTONE .INP_BUF[6+4] OF
    SET
    ['R','T','A','I']: 0;
    [OTHERWISE]:      SOR_ERROR(SRTTRN$_INV_PROCESS);
    TES;

SELECTONE .INP_BUF[25] OF
    SET
    [' ']:          0;
    ['E']:          EBCDIC = TRUE;
    ['X']:          ALTSEQ = TRUE;
    [OTHERWISE]:    SOR_ERROR(SRTTRN$_INV_ALT);
    TES;

IF NOT CVT_DTB(4, INP_BUF[28], OUTLRL) THEN OUTLRL = %X'FFFF';
END;

GET_LINE();

END;

```

```
.PSECT $SPLITS,NOWRT,NOEXE,2
```

54 52 4F 53 0009C P.AAS: .ASCII \SORT\

```
.PSECT $CODE$,NOWRT,2
```

		001C 00000 DO_HEADER:				
	54	FB1C	CF 9E 00002	.WORD	Save R2,R3,R4	: 1269
	53	0000'	CF 9E 00007	MOVAB	SOR_ERROR, R4	: :
	52	0000'	CF 9E 0000C	MOVAB	INP_BUF+10, R3	: :
	01	28	A2 D1 00011	MOVAB	ASCENDING, R2	: :
			03 13 00015	CPL	PASS, #1	: 1302
		00CC	31 00017	BEQL	1\$	: :
		F6	A3 9F 0001A 1\$:	BRW	15\$	: :
		0182	C3 DD 0001D	PUSHAB	INP_BUF	: 1310
0000V	CF		02 FB 00021	PUSHL	INP_REC_SIZ	: :
	50	07	A3 9A 00026	CALLS	#2, -UPCASE	: :
	20		50 91 0002A	MOVZBL	INP_BUF+17, R0	: 1314
			06 13 0002D	CMPB	R0, #32	: 1316
			50 91 0002F	BEQL	2\$	: :
41	8F		05 12 00033	CMPB	R0, #65	: :
				BNEQ	3\$	: :

	62		01	D0	00035	2\$:	MOVL	#1, ASCENDING		
			13	11	00038		BRB	5\$		
44	8F		50	91	0003A	3\$:	CMPB	R0, #68	1317	
			04	12	0003E		BNEQ	4\$		
			62	D4	00040		CLRL	ASCENDING		
			09	11	00042		BRB	5\$		
		001C8044	8F	DD	00044	4\$:	PUSHL	#1867844	1318	
	64		01	FB	0004A		CALLS	#1, SOR_ERROR		
	50	11	A3	9A	0004D	5\$:	MOVZBL	INP_BUF+27, R0	1323	
	20		50	91	00051		CMPB	R0, #32	1325	
			06	12	00054		BNEQ	6\$		
34	A2		01	D0	00056		MOVL	#1, NOSTRIP		
			14	11	0005A		BRB	8\$		
58	8F		50	91	0005C	6\$:	CMPB	R0, #88	1326	
			05	12	00060		BNEQ	7\$		
		34	A2	D4	00062		CLRL	NOSTRIP		
			09	11	00065		BRB	8\$		
		001C804C	8F	DD	00067	7\$:	PUSHL	#1867852	1327	
	64		01	FB	0006D		CALLS	#1, SOR_ERROR		
FC	A3	0000'	CF	D1	00070	8\$:	CMPL	P.AAS, INP_BUF+6	1330	
			06	13	00076		BEQL	9\$		
2C	A2		20	D0	00078		MOVL	#32, PROCESS	1332	
			04	11	0007C		BRB	10\$		
2C	A2		63	9A	0007E	9\$:	MOVZBL	INP_BUF+10, PROCESS	1334	
	50		63	9A	00082	10\$:	MOVZBL	INP_BUF+10, R0	1336	
41	8F		50	91	00085		CMPB	R0, #65	1338	
			1B	13	00089		BEQL	11\$		
49	8F		50	91	0008B		CMPB	R0, #73		
			15	13	0008F		BEQL	11\$		
52	8F		50	91	00091		CMPB	R0, #82		
			0F	13	00095		BEQL	11\$		
54	8F		50	91	00097		CMPB	R0, #84		
			09	13	0009B		BEQL	11\$		
		001C8032	8F	DD	0009D		PUSHL	#1867826	1339	
	64		01	FB	000A3		CALLS	#1, SOR_ERROR		
	50	0F	A3	9A	000A6	11\$:	MOVZBL	INP_BUF+25, R0	1342	
	20		50	91	000AA		CMPB	R0, #32	1344	
			21	13	000AD		BEQL	14\$		
45	8F		50	91	000AF		CMPB	R0, #69	1345	
			06	12	000B3		BNEQ	12\$		
30	A2		01	D0	000B5		MOVL	#1, EBCDIC		
			15	11	000B9		BRB	14\$		
58	8F		50	91	000BB	12\$:	CMPB	R0, #88	1346	
			06	12	000BF		BNEQ	13\$		
04	A2		01	D0	000C1		MOVL	#1, ALTSEQ		
			09	11	000C5		BRB	14\$		
		001C8004	8F	DD	000C7	13\$:	PUSHL	#1867780	1347	
	64		01	FB	000CD		CALLS	#1, SOR_ERROR		
		38	A2	9F	000D0	14\$:	PUSHAB	OUTLRL	1350	
		12	A3	9F	000D3		PUSHAB	INP_BUF+28		
			04	DD	000D6		PUSHL	#4		
0000V	CF		03	FB	000D8		CALLS	#3, CVT_DTB		
	06		50	EB	000DD		BLBS	R0, 15\$		
38	A2	FFFF	8F	3C	000E0		MOVZWL	#65535, OUTLRL		
0000V	CF		00	FB	000E6	15\$:	CALLS	#0, GET_LINE	1353	
			04	00	000EB		RET		1355	

SRTTRN
V04-000

E 15
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 44
(14)

; Routine Size: 236 bytes, Routine Base: \$CODE\$ + 04DE


```
: 1273      1356 1 ROUTINE CVT_OTB(X: REF VECTOR[,BYTE]) =
: 1274      1357 2 BEGIN
: 1275      1358 2 LOCAL
: 1276      1359 2 PTR, RES, NUM;
: 1277      1360 2 PTR = X[0];
: 1278      1361 2 RES = 0;
: 1279      1362 2 DECR I FROM 2 TO 0 DO
: 1280      1363 3 BEGIN
: 1281      1364 3 NUM = CH$RCHAR_A(PTR) - %C'0';
: 1282      1365 3 IF .NUM GEQU 8
: 1283      1366 3 THEN
: 1284      1367 4 BEGIN
: 1285      1368 4 SOR_ERROR(SRTTRN$_INV_ALT);
: 1286      1369 4 RETURN 0;
: 1287      1370 3 END;
: 1288      1371 3 RES = .RES * 8 + .NUM;
: 1289      1372 2 END;
: 1290      1373 2 RETURN .RES<0,8,0>; ! Just use 8 bits
: 1291      1374 1 END;
```

			003C	00000	CVT_OTB: .WORD	Save R2,R3,R4,R5		1356
55	04	AC	D0	00002	MOVL	X, PTR		1360
		53	D4	00006	CLRL	RES		1361
54		02	D0	00008	MOVL	#2, I		1362
50		85	9A	0000B	1\$: MOVZBL	(PTR)+, R0		1364
52	D0	A0	9E	0000E	MOVAB	-48(R0), NUM		
08		52	D1	00012	CMPL	NUM, #8		1365
		0D	1F	00015	BLSSU	2\$		
	001C8004	8F	DD	00017	PUSHL	#1867780		1368
FA14	CF	01	FB	0001D	CALLS	#1, SOR_ERROR		
		0B	11	00022	BRB	3\$		1369
53		6243	7E	00024	2\$: MOVAQ	(NUM)[RES], RES		1371
E0		54	F4	00028	SOBGEQ	I, 1\$		1362
50		53	9A	0002B	MOVZBL	RES, R0		1373
			04	0002E	RET			
		50	D4	0002F	3\$: CLRL	R0		1374
			04	00031	RET			

; Routine Size: 50 bytes, Routine Base: \$CODE\$ + 05CA

```
: 1293      1375 1 ROUTINE CVT_BTO(X) =  
: 1294      1376 2 BEGIN  
: 1295      1377 2 OWN  
: 1296      1378 2 O: VECTOR[4,BYTE];  
: 1297      1379 2 LOCAL  
: 1298      1380 2 Y;  
: 1299      1381 2  
: 1300      1382 2 Y = .X;  
: 1301      1383 2 O[2] = .Y MOD 8 OR %C'0'; Y = .Y / 8;  
: 1302      1384 2 O[1] = .Y MOD 8 OR %C'0'; Y = .Y / 8;  
: 1303      1385 2 O[0] = .Y MOD 8 OR %C'0';  
: 1304      1386 2  
: 1305      1387 2 RETURN O[0];  
: 1306      1388 2  
: 1307      1389 1 END;
```

.PSECT \$OWNS\$,NOEXE,2

00778 0: .BLKB 4

.PSECT \$CODE\$,NOWRT,2

			52	0000'	CF 9E 00002	CVT_BTO: .WORD	Save R2	: 1375
			51	04	AC D0 00007	MOVAB	O, R2	: 1382
7E		00	51		01 7A 0000B	MOVL	X, Y	: 1383
50		50	8E		08 7B 00010	EMUL	#1, Y, #0, -(SP)	
	02	A2	50		30 89 00015	EDIV	#8, (SP)+, R0, R0	
			51		08 C6 0001A	BISB3	#48, R0, 0+2	
7E		00	51		01 7A 0001D	DIVL2	#8, Y	: 1384
50		50	8E		08 7B 00022	EMUL	#1, Y, #0, -(SP)	
	01	A2	50		30 89 00027	EDIV	#8, (SP)+, R0, R0	
			51		08 C6 0002C	BISB3	#48, R0, 0+1	
7E		00	51		01 7A 0002F	DIVL2	#8, Y	: 1385
50		50	8E		08 7B 00034	EMUL	#1, Y, #0, -(SP)	
		62	50		30 89 00039	EDIV	#8, (SP)+, R0, R0	
			50		62 9E 0003D	BISB3	#48, R0, 0	: 1387
					04 00040	MOVAB	O, R0	: 1389
						RET		

; Routine Size: 65 bytes, Routine Base: \$CODE\$ + 05FC


```
1309 1390 1 ROUTINE DO_ALTSEQ: NOVALUE =
1310 1391 1
1311 1392 1 ++
1312 1393 1
1313 1394 1 FUNCTIONAL DESCRIPTION:
1314 1395 1
1315 1396 1     Translates an alternate collating sequence line.
1316 1397 1
1317 1398 1 FORMAL PARAMETERS:
1318 1399 1
1319 1400 1     NONE
1320 1401 1
1321 1402 1 IMPLICIT INPUTS:
1322 1403 1
1323 1404 1     NONE
1324 1405 1
1325 1406 1 IMPLICIT OUTPUTS:
1326 1407 1
1327 1408 1     NONE
1328 1409 1
1329 1410 1 ROUTINE VALUE:
1330 1411 1
1331 1412 1     0 or 1
1332 1413 1
1333 1414 1 SIDE EFFECTS:
1334 1415 1
1335 1416 1     NONE
1336 1417 1
1337 1418 1 NOTES:
1338 1419 1
1339 1420 1     There is a basic difference between how collating values are specified
1340 1421 1     in the old format spec file and the new format.  In the old format,
1341 1422 1     oct1 = oct2 means that:
1342 1423 1         character oct1 is given the collating value oct2,
1343 1424 1     while in the new format, oct1 = oct2 means:
1344 1425 1         character oct1 is given the collating value of character oct2.
1345 1426 1
1346 1427 1     In order to translate this, we maintain a table representing the old
1347 1428 1     collating sequence table.  When we make a modification, we search for
1348 1429 1     a character that has that collating value (or is close), and make the
1349 1430 1     appropriate modification.
1350 1431 1 --
1351 1432 2 BEGIN
1352 1433 2 LOCAL
1353 1434 2     SEQ: VECTOR[256, BYTE],
1354 1435 2     PTR,
1355 1436 2     FIRST,
1356 1437 2     PAIR_PTR: REF VECTOR[BYTE];
1357 1438 2 MACRO
1358 1439 2     STR_FIRST =
1359 1440 2     %STRING('/',X_COLL,'=(',X_SEQU,':',X_ASC,', ',X_MODF,':(') %;
1360 1441 2
1361 1442 2 IF .PASS NEQ 2
1362 1443 2 THEN
1363 1444 2     BEGIN
1364 1445 2     DO GET_LINE() UNTIL .LINE_TYPE NEQ ALTSEQ_LINE;
1365 1446 2     RETURN;
```

```
1366 1447 2
1367 1448 2
1368 1449 2
1369 1450 2
1370 1451 2
1371 1452 2
1372 1453 2
1373 1454 2
1374 1455 2
1375 1456 2
1376 1457 2
1377 1458 2
1378 1459 2
1379 1460 2
1380 1461 2
1381 1462 2
1382 1463 2
1383 1464 2
1384 1465 2
1385 1466 2
1386 1467 2
1387 1468 2
1388 1469 2
1389 1470 2
1390 1471 2
1391 1472 2
1392 1473 2
1393 1474 2
1394 1475 2
1395 1476 2
1396 1477 2
1397 1478 2
1398 1479 2
1399 1480 2
1400 1481 2
1401 1482 2
1402 1483 2
1403 1484 2
1404 1485 2
1405 1486 2
1406 1487 2
1407 1488 2
1408 1489 2
1409 1490 2
1410 1491 2
1411 1492 2
1412 1493 2
1413 1494 2
1414 1495 2
1415 1496 2
1416 1497 2
1417 1498 2
1418 1499 2
1419 1500 2
1420 1501 2
1421 1502 2
1422 1503 2
```

```
END;
! Initialize the collating sequence table
DECR I FROM 255 TO 0 DO SEQ[I] = .I;
! Check that user defined sequence has been requested
IF NOT .ALTSEQ
THEN
    SOR_ERROR(SRTTRN$_UNIX_ALTSEQ);
! Indicate that we haven't written any text yet
FIRST = TRUE;
! While there are altseq lines
WHILE TRUE DO
    BEGIN
        ! Look for max of groups of 6 octal digits
        INCRA PAIR_PTR FROM INP_BUF[8] TO INP_BUF[74] BY 6 DO
            BEGIN
                MAP
                PAIR_PTR: REF VECTOR[.BYTE];
                LOCAL
                CH1, CH2,
                IX: REF VECTOR[.BYTE];
                ! Should not be any blanks in next 6 columns
                IF NOT CH$FAIL(CH$FIND_CH(6, .PAIR_PTR, %C' ')) THEN EXITLOOP;
                ! Fetch the two numbers
                CH1 = CVT_OTB(PAIR_PTR[0]);
                CH2 = CVT_OTB(PAIR_PTR[3]);
                ! Find a character that has nearly the same collating value
                DECR I FROM .CH2 TO 0 DO
                    BEGIN
                        IX = CH$FIND_CH( 256, SEQ[0], .I );
                        IF NOT CH$FAIL(.IX) THEN EXITLOOP;
                    END;
                IF CH$FAIL(.IX) THEN
                    INCR I FROM .CH2+1 TO 255 DO
                        BEGIN
                            IX = CH$FIND_CH( 256, SEQ[0], .I );
                            IF NOT CH$FAIL(.IX) THEN EXITLOOP;
                        END;
                IF .FIRST
                THEN PUT_TEXT(LENADR_(STR_FIRST))
                ELSE PUT_TEXT(LENADR_(' ', ' '));
```



```
: 1423      1504  4      FIRST = FALSE;
: 1424      1505  4      PUT_TEXT(LENADR ('%0'),
: 1425      1506  4      -3, PAIR_PTR[0],
: 1426      1507  5      %CHARCOUNT(' ? %0'), (SELECTONE SIGN(.CH2 - CH$RCHAR(.IX)) OF
: 1427      1508  5      SET
: 1428      1509  5      [-1]: UPLIT BYTE(' < %0');
: 1429      1510  5      [ 0]: UPLIT BYTE(' = %0');
: 1430      1511  5      [+1]: UPLIT BYTE(' > %0');
: 1431      1512  4      TES);
: 1432      1513  4      3. CVT_BTO(CH$DIFF(.IX, SEQ[0]));
: 1433      1514  4
: 1434      1515  4      ! Update the collating sequence table
: 1435      1516  4      !
: 1436      1517  4      SEQ[.CH1] = .CH2;
: 1437      1518  3      END;
: 1438      1519  3
: 1439      1520  3      ! Look for next altseq
: 1440      1521  3      !
: 1441      1522  3      GET_LINE();
: 1442      1523  3
: 1443      1524  3      IF .LINE_TYPE NEQ ALTSEQ_LINE THEN EXITLOOP;
: 1444      1525  3
: 1445      1526  2      END;
: 1446      1527  2
: 1447      1528  2      ! Add end parentheses; through processing the altseq
: 1448      1529  2      !
: 1449      1530  2      IF .FIRST
: 1450      1531  2      THEN PUT_TEXT(LENADR ('')));
: 1451      1532  2      ELSE PUT_TEXT(LENADR_ ('')));
: 1452      1533  2
: 1453      1534  1      END;
```

```
55 51 45 53 5F 47 4E 49 54 41 4C 4C 4F 43 2F 000A0 P.AAT: .PSECT $SPLITS,NOWRT,NOEXE,2
3A 45 43 4E 45 55 51 45 53 28 3D 45 43 4E 45 .ASCII \COLLATING_SEQUENCE=(SEQUENCE:ASCII, MOD\
20 28 3A 4E 4F 4D 20 2C 49 49 43 53 41 000AF
4F 25 20 3C 20 000BE
4F 25 20 3D 20 000C8
4F 25 20 3E 20 000D4 P.AAU: .ASCII \IFICATION:( \
29 29 000D6 P.AAV: .ASCII \
29 29 000D8 P.AAW: .ASCII \ < %0\
29 29 000DD P.AAX: .ASCII \ = %0\
29 29 000E2 P.AAY: .ASCII \ > %0\
29 29 000E7 P.AAZ: .ASCII \)\
29 29 000E8 P.ABA: .ASCII \)\)
```

```
58 0000V CF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 1390
5A 0000' CF 9E 00007 MOVAB PUT_TEXT, R11
59 0000' CF 9E 0000C MOVAB LINE_TYPE, R10
MOVAB P.AAT, R9
```

	5E	FF00	CE	9E	00011	MOVAB	-256(SP), SP		
	02	14	AA	D1	00016	CMPL	PASS, #2		1442
			0B	13	0001A	BEQL	2\$		
0000V	CF		00	FB	0001C	CALLS	#0, GET_LINE		1445
	02		6A	D1	00021	CMPL	LINE_TYPE, #2		
			F6	13	00024	BEQL	1\$		
				04	00026	RET			1444
	50	FF	8F	9A	00027	MOVZBL	#255, I		1451
6E40	F9		50	90	0002B	MOVB	I, SEQ[1]		
	0B		50	F4	0002F	SOBGEQ	I, 3\$		
		F0	AA	E8	00032	BLBS	ALTSEQ, 4\$		1455
F982	CF	001C8078	8F	DD	00036	PUSHL	#1867896		1457
	57		01	FB	0003C	CALLS	#1, SOR_ERROR		
	54	0000'	01	D0	00041	MOVL	#1, FIRST		1461
	58	0000'	CF	9E	00044	MOVAB	INP_BUF+8, R4		1470
			CF	9E	00049	MOVAB	INP_BUF+74, R8		
64	06		00CE	31	0004E	BRW	24\$		
			20	3A	00051	LOCC	#32, #6, (PAIR_PTR)		1480
			02	12	00055	BNEQ	7\$		
			51	D4	00057	CLRL	R1		
			51	D5	00059	TSTL	R1		
			03	13	0005B	BEQL	8\$		
			00C7	31	0005D	BRW	25\$		
			54	DD	00060	PUSHL	PAIR_PTR		1484
FF26	CF		01	FB	00062	CALLS	#1, CVT_OTB		
	55		50	D0	00067	MOVL	R0, CH1		
		03	A4	9F	0006A	PUSHAB	3(PAIR_PTR)		1485
FF1B	CF		01	FB	0006D	CALLS	#1, CVT_OTB		
	56		50	D0	00072	MOVL	R0, CH2		
	52		01	A6	9E	00075	MOVAB	1(R6), I	1489
			0F	11	00079	BRB	11\$		
6E	0100		52	3A	0007B	LOCC	I, #256, SEQ		1491
			02	12	00081	BNEQ	10\$		
			51	D4	00083	CLRL	R1		
	53		51	D0	00085	MOVL	R1, IX		
			03	12	00088	BNEQ	12\$		1492
	EE		52	F4	0008A	SOBGEQ	I, 9\$		1489
			53	D5	0008D	TSTL	IX		1494
			1C	12	0008F	BNEQ	16\$		
	52		56	D0	00091	MOVL	CH2, I		1495
			0F	11	00094	BRB	15\$		
6E	0100		52	3A	00096	LOCC	I, #256, SEQ		1497
			02	12	0009C	BNEQ	14\$		
			51	D4	0009E	CLRL	R1		
	53		51	D0	000A0	MOVL	R1, IX		
			08	12	000A3	BNEQ	16\$		1498
E9	52	000000FF	8F	F3	000A5	AOBLEQ	#255, I, 13\$		1495
	06		57	E9	000AD	BLBC	FIRST, 17\$		1501
			59	DD	000B0	PUSHL	R9		1502
			34	DD	000B2	PUSHL	#52		
			05	11	000B4	BRB	18\$		
		34	A9	9F	000B6	PUSHAB	P.AAU		1503
			02	DD	000B9	PUSHL	#2		
	6B		02	FB	000BB	CALLS	#2, PUT_TEXT		
			57	D4	000BE	CLRL	FIRST		1504
	50		6E	9E	000C0	MOVAB	SEQ, R0		1513
7E	53		50	C3	000C3	SUBL3	R0, IX, -(SP)		

	FEF3	CF	01	FB	000C7	CALLS	#1, CVT_BT0	
			50	DD	000CC	PUSHL	R0	
			03	DD	000CE	PUSHL	#3	1506
		51	63	9A	000D0	MOVZBL	(IX), R1	1507
		51	56	C2	000D3	SUBL2	CH2, R1	
			51	D5	000D6	TSTL	R1	
			51	DC	000D8	MOVPSL	R1	
51		02	02	EF	000DA	EXTZV	#2, #2, R1, R1	
			51	D7	000DF	DECL	R1	
	FFFFFFF	8F	51	D1	000E1	CMPL	R1, #-1	1509
			06	12	000E8	BNEQ	19\$	
		50	38	A9	9E 000EA	MOVAB	P.AAW, R0	
			18	11	000EE	BRB	22\$	
			51	D5	000F0	TSTL	R1	1510
			06	12	000F2	BNEQ	20\$	
		50	3D	A9	9E 000F4	MOVAB	P.AAX, R0	
			0E	11	000F8	BRB	22\$	
		01	51	D1	000FA	CMPL	R1, #1	1511
			05	13	000FD	BEQL	21\$	
		7E	01	CE	000FF	MNEGL	#1, -(SP)	
			06	11	00102	BRB	23\$	
		50	42	A9	9E 00104	MOVAB	P.AAY, R0	
			50	DD	00108	PUSHL	R0	
			05	DD	0010A	PUSHL	#5	1506
			54	DD	0010C	PUSHL	PAIR_PTR	
			03	DD	0010E	PUSHL	#3	
			36	A9	9F 00110	PUSHAB	P.AAV	1505
			02	DD	00113	PUSHL	#2	1506
	6B		08	FB	00115	CALLS	#8, PUT_TEXT	
	6E45		56	90	00118	MOVB	CH2, SEQ[CH1]	1517
	54		06	C0	0011C	ADDL2	#6, PAIR_PTR	1470
	58		54	D1	0011F	CMPL	PAIR_PTR, R8	
			03	1A	00122	BGTRU	25\$	
			FF2A	31	00124	BRW	6\$	
	0000V	CF	00	FB	00127	CALLS	#0, GET_LINE	1522
		02	6A	D1	0012C	CMPL	LINE_TYPE, #2	1524
			03	12	0012F	BNEQ	26\$	
			FF10	31	00131	BRW	5\$	
		07	57	E9	00134	BLBC	FIRST, 27\$	1530
			47	A9	9F 00137	PUSHAB	P.AAZ	1531
			01	DD	0013A	PUSHL	#1	
			05	11	0013C	BRB	28\$	
			48	A9	9F 0013E	PUSHAB	P.ABA	1532
			02	DD	00141	PUSHL	#2	
	6B		02	FB	00143	CALLS	#2, PUT_TEXT	
			04	00146	RET			1534

; Routine Size: 327 bytes, Routine Base: \$CODE\$ + 063D

```
: 1455      1535 1 ROUTINE PUT_NEST(NESTED): NOVALUE =  
: 1456      1536 2   BEGIN  
: 1457      1537 2   IF .NESTED  
: 1458      1538 2   THEN  
: 1459      1539 3       BEGIN  
: 1460      1540 3       PUT_TEXT(LENADR_(' '));  
: 1461      1541 3       PUT_LINE(1);  
: 1462      1542 3       END  
: 1463      1543 2   ELSE  
: 1464      1544 2       PUT_TEXT(LENADR_(' / '));  
: 1465      1545 1   END;
```

.PSECT \$SPLITS,NOWRT,NOEXE,2

```
2C 000EA P.ABB: .ASCII \,\  
2F 000EB P.ABC: .ASCII \/\
```

.PSECT \$CODE\$,NOWRT,2

```
0000 00000 PUT_NEST:  
      13      04 AC E9 00002 .WORD Save nothing  
      0000' CF 9F 00006 BLBC NESTED, 1$  
0000V CF 01 DD 0000A PUSHAB P.ABB  
0000V CF 02 FB 0000C PUSHBL #1  
0000V CF 01 DD 00011 CALLS #2, PUT_TEXT  
0000V CF 01 FB 00013 CALLS #1, PUT_LINE  
      0000' CF 04 00018 RET  
0000V CF 01 DD 0001D 1$: PUSHAB P.ABC  
0000V CF 02 FB 0001F PUSHBL #1  
      04 00024 CALLS #2, PUT_TEXT  
RET
```

; Routine Size: 37 bytes, Routine Base: \$CODE\$ + 0784

SRTTRN
V04-000

N 15
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 53
(19)

```

: 1467      1546 1 ROUTINE CHECK_LENGTH(NAM: REF FLD_BLOCK): NOVALUE =
: 1468      1547 2 BEGIN
: 1469      1548 2 LOCAL
: 1470      1549 2 SIZ;
: 1471      1550 2
: 1472      1551 2 SIZ = .NAME[FLD_SIZ];
: 1473      1552 2
: 1474      1553 2 !<s1>
: 1475      1554 2 SELECTONE .NAME[FLD_DTY] OF
: 1476      1555 2 SET
: 1477      1556 2 [DSC$K_DTYPE_NL
: 1478      1557 2 DSC$K_DTYPE_NR]: SIZ = .SIZ + 1;
: 1479      1558 2 [DSC$K_DTYPE_P]: SIZ = .SIZ / 2 + 1;
: 1480      1559 2 [OTHERWISE]: 0;
: 1481      1560 2 TES;
: 1482      1561 2
: 1483      1562 2 FMT_CURR[FMT_OUTLRL] = .FMT_CURR[FMT_OUTLRL] + .SIZ;
: 1484      1563 2 IF .FMT_CURR[FMT_OUTLRL] GTRU .OUTLRL
: 1485      1564 2 THEN
: 1486      1565 2 BEGIN
: 1487      1566 2 SOR_ERROR(SRTTRNS_DATALONG);
: 1488      1567 2 FMT_CURR[FMT_OUTLRL] = 0;
: 1489      1568 2 END;
: 1490      1569 2
: 1491      1570 1 END;
```

0000 00000 CHECK_LENGTH:							
	51	04	AC	D0 00002	.WORD	Save nothing	: 1546
	50	0A	A1	3C 00006	MOVL	NAM, R1	: 1551
	51	10	A1	9A 0000A	MOVZWL	10(R1), SIZ	
	10		51	91 0000E	MOVZBL	16(R1), R1	: 1554
			05	13 00011	CMPB	R1, #16	: 1556
	12		51	91 00013	BEQL	1\$	
			04	12 00016	CMPB	R1, #18	
			50	D6 00018 1\$:	BNEQ	2\$	
			0D	11 0001A	INCL	SIZ	: 1557
	15		51	91 0001C 2\$:	BRB	3\$	
			08	12 0001F	CMPB	R1, #21	: 1558
			02	C7 00021	BNEQ	3\$	
51	50		A1	9E 00025	DIVL3	#2, SIZ, R1	
	50	01	A1	9E 00025	MOVAB	1(R1), SIZ	
	51	0000'	CF	D0 00029 3\$:	MOVL	FMT_CURR, R1	: 1562
0C	A1		50	C0 0002E	ADDL2	SIZ, 12(R1)	
0000'	CF	0C	A1	D1 00032	CMPL	12(R1), OUTLRL	: 1563
			13	1B 00038	BLEQU	4\$	
		001C8098	8F	DD 0003A	PUSHL	#1867928	: 1566
F812	CF		01	FB 00040	CALLS	#1, SOR_ERROR	
	50	0000'	CF	D0 00045	MOVL	FMT_CURR, R0	: 1567
		0C	A0	D4 0004A	CLRL	12(R0)	
			04	0004D 4\$:	RET		: 1570

; Routine Size: 78 bytes, Routine Base: \$CODE\$ + 07A9

B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I

SRTTRN
V04-000

B 16
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 BLISS-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 54
(19)


```
: 1493      1571 1 ROUTINE DO_FIELD(NESTED): NOVALUE =
: 1494      1572 1
: 1495      1573 1 !++
: 1496      1574 1
: 1497      1575 1 FUNCTIONAL DESCRIPTION:
: 1498      1576 1
: 1499      1577 1     Does all the field lines in specification file which
: 1500      1578 1     does not have an include or omit line.
: 1501      1579 1
: 1502      1580 1 FORMAL PARAMETERS:
: 1503      1581 1
: 1504      1582 1     NESTED True to indicate that the field definition is nested
: 1505      1583 1     within an omit/include.
: 1506      1584 1
: 1507      1585 1 IMPLICIT INPUTS:
: 1508      1586 1
: 1509      1587 1     NONE
: 1510      1588 1
: 1511      1589 1 IMPLICIT OUTPUTS:
: 1512      1590 1
: 1513      1591 1     NONE
: 1514      1592 1
: 1515      1593 1 ROUTINE VALUE:
: 1516      1594 1
: 1517      1595 1     0 or 1
: 1518      1596 1
: 1519      1597 1 SIDE EFFECTS:
: 1520      1598 1
: 1521      1599 1     NONE
: 1522      1600 1 --
: 1523      1601 2 BEGIN
: 1524      1602 2 LOCAL
: 1525      1603 2 LEN;
: 1526      1604 2
: 1527      1605 2 ! Look for all fields
: 1528      1606 2 !
: 1529      1607 2 WHILE .LINE_TYPE EQL FIELD_LINE DO
: 1530      1608 3 BEGIN
: 1531      1609 3
: 1532      1610 3 ! Convert the first 8 character position to upper case
: 1533      1611 3 !
: 1534      1612 3 UPCASE(8, INP_BUF);
: 1535      1613 3
: 1536      1614 3 ! Force, key, or data field
: 1537      1615 3 !
: 1538      1616 3 SELECTONE .INP_BUF[6] OF
: 1539      1617 3 SET
: 1540      1618 3 ['F']: ! Force field
: 1541      1619 4 BEGIN
: 1542      L 1620 4 %if k_new
: 1543      1621 4 %then
: 1544      1622 4 LOCAL
: 1545      1623 4 FFLD: REF FFLD_BLOCK;
: 1546      1624 4
: 1547      1625 4 IF .PASS EQL 1
: 1548      1626 4 THEN
: 1549      1627 5 BEGIN
```



```
: 1550      1628  5
: 1551      1629  5
: 1552      1630  5
: 1553      1631  5
: 1554      1632  5
: 1555      1633  5
: 1556      1634  5
: 1557      1635  5
: 1558      1636  5
: 1559      1637  5
: 1560      1638  4
: 1561      U 1639  4 %else
: 1562      1640  4 %fi
: 1563      1641  4
: 1564      1642  4
: 1565      1643  4
: 1566      1644  4
: 1567      1645  4
: 1568      1646  4
: 1569      1647  4
: 1570      1648  4
: 1571      1649  5
: 1572      L 1650  5 %if k_new
: 1573      1651  5 %then
: 1574      1652  5
: 1575      1653  5
: 1576      1654  5
: 1577      U 1655  5 %else
: 1578      U 1656  5
: 1579      U 1657  5
: 1580      U 1658  5
: 1581      U 1659  5
: 1582      U 1660  5
: 1583      U 1661  5
: 1584      U 1662  5
: 1585      1663  5 %fi
: 1586      1664  5
: 1587      1665  5
: 1588      1666  4
: 1589      1667  5
: 1590      1668  5
: 1591      1669  5
: 1592      1670  5
: 1593      1671  5
: 1594      1672  5
: 1595      1673  5
: 1596      1674  5
: 1597      1675  5
: 1598      1676  5
: 1599      1677  5
: 1600      L 1678  5 %if k_new
: 1601      1679  5 %then
: 1602      U 1680  5 %else
: 1603      U 1681  5
: 1604      U 1682  5
: 1605      U 1683  5
: 1606      U 1684  5

      |
      | Create a KEY_BLOCK for this force field
      |
      | LOCAL FLD: REF FLD_BLOCK;
      | GET_VM(FLD_K_SIZE, FLD);
      | FLD[FLD_POS] = -1;
      | FLD[FLD_SIZE] = 1;
      | FLD[FLD_DTY] = DSC$K_DTYPE_BU;
      | BBLOCK[NEW KEY(FLD[BASE]), KEY_ASCE] = .ASCENDING;
      | FFLD = FLD[FLD_FFLD] - %FIELDEXPAND(FFLD_NEXT, 0);
      | END;

      IF .INP_BUF[18] NEQ %C' '
      THEN
        IF .PASS EQL 3 THEN SOR_ERROR(SRTTRN$_UNX_FORCE);
      ! Unconditional?
      !
      IF .INP_BUF[16] EQL %C' '
      THEN
        BEGIN
          IF .PASS EQL 1
          THEN
            FFLD = FFLD[FFLD_NEXT] = GET_FFLD(.INP_BUF[17], 0);
          IF .PASS EQL 3
          THEN
            BEGIN
              PUT_NEST(.NESTED);
              PUT_TEXT(LENADR(X_KEY, '%0'),
                3, CVT_BTO(.INP_BUF[17]));
            END;
          GET_LINE();
          END
        ELSE
          BEGIN
            |
            | Lots of fun.
            | We have a conditional force field.
            | On the first pass:
            |   Define the field.
            | On the second pass:
            |   Describe the condition(s).
            | On the third pass:
            |   Describe the field.
            |
          END
        END
      IF .PASS EQL 3
      THEN
        BEGIN
          PUT_NEST(.NESTED);
```



```
: 1607      U 1685 5      PUT_TEXT( LENADR_(X_KEY,'=(',%CHAR(9)) );
: 1608      U 1686 5      END;
: 1609      1687 5 %fi
: 1610      1688 5
: 1611      1689 6      WHILE TRUE DO
: 1612      1690 6      BEGIN
: 1613      1691 6      LOCAL NAM, POS;
: 1614      1692 6
: 1615      1693 6      IF NOT CVT_DTB(4, INP_BUF[12], POS)
: 1616      1694 6      THEN
: 1617      1695 6          IF .PASS EQL 2
: 1618      1696 6          THEN
: 1619      1697 6              SOR_ERROR(SRTTRN$_NO_FORCE_TO);
: 1620      1698 6
: 1621      1699 6      NAM = TREE_INSERT(.POS, 1, DSC$K_DTYPE_BU);
: 1622      1700 6      SELECTONE .PASS OF
: 1623      L 1701 6 %if k_new
: 1624      L 1702 6 %then
: 1625      1703 6
: 1626      1704 7
: 1627      1705 7
: 1628      1706 7
: 1629      1707 6
: 1630      U 1708 6 %else
: 1631      U 1709 6
: 1632      1710 6 %fi
: 1633      1711 6
: 1634      1712 7
: 1635      1713 7
: 1636      1714 7
: 1637      1715 7
: 1638      1716 7
: 1639      1717 7
: 1640      1718 7
: 1641      1719 7
: 1642      1720 7
: 1643      1721 7
: 1644      1722 6
: 1645      L 1723 6 %if k_new
: 1646      L 1724 6 %then
: 1647      1725 6
: 1648      U 1726 6 %else
: 1649      U 1727 6
: 1650      U 1728 6
: 1651      U 1729 6
: 1652      U 1730 6
: 1653      U 1731 6
: 1654      U 1732 6
: 1655      U 1733 6
: 1656      U 1734 6
: 1657      U 1735 6
: 1658      U 1736 6
: 1659      U 1737 6
: 1660      U 1738 6
: 1661      U 1739 6
: 1662      U 1740 6
: 1663      1741 6 %fi
```

```
PUT_TEXT( LENADR_(X_KEY,'=(',%CHAR(9)) );
END;

WHILE TRUE DO
BEGIN
LOCAL NAM, POS;

IF NOT CVT_DTB(4, INP_BUF[12], POS)
THEN
IF .PASS EQL 2
THEN
SOR_ERROR(SRTTRN$_NO_FORCE_TO);

NAM = TREE_INSERT(.POS, 1, DSC$K_DTYPE_BU);
SELECTONE .PASS OF
SET

[1]:
BEGIN
FFLD = FFLD[FFLD_NEXT] = GET_FFLD(.INP_BUF[17],
COND_NUM = .COND_NUM+1);
END;

[1]: 0;

[2]:
BEGIN
PUT_TEXT(
LENADR_('/' X COND,'=(NAME:COND ')
3, CVT_BTO(COND_NUM = .COND_NUM+ 1),
LENADR_(' ' X TEST,':( '));
PUT_NAME(.NAM);
PUT_TEXT(
LENADR_(' EQ %0'),
3, CVT_BTO(.INP_BUF[16]),
LENADR_(' '));
END;

[3]: 0;

[3]:
BEGIN
Note that we can't use CVT_BTO twice here
PUT_TEXT(
LENADR_(X IF,' COND '),
3, CVT_BTO(COND_NUM = .COND_NUM + 1));
PUT_TEXT(
LENADR_(' ' X THEN,' %0'),
3, CVT_BTO(.INP_BUF[17]),
LENADR_(' ' X ELSE));
PUT_LINE(.NESTED+1);
END;
```



```
: 1664      1742  6      TES;
: 1665      1743  6      GET_LINE();
: 1666      1744  6      UPCASE(8, INP_BUF);
: 1667      1745  6      ! Is next line a continuation of force key?
: 1668      1746  6      !
: 1669      1747  6      !
: 1670      1748  6      IF .LINE_TYPE EQL FIELD_LINE AND
: 1671      1749  6      .INP_BUF[6] EQL %C'F' AND
: 1672      1750  6      .INP_BUF[18] NEQ %C' '
: 1673      1751  6      THEN
: 1674      1752  6      IF .INP_BUF[16] EQL %C' '
: 1675      1753  6      THEN NAM = .INP_BUF[17]
: 1676      1754  6      ELSE NAM = -1
: 1677      1755  6      ELSE
: 1678      1756  6      IF .ASCENDING
: 1679      1757  6      THEN NAM = %X'FF'
: 1680      1758  6      ELSE NAM = %X'00';
: 1681      1759  6      IF .NAM GEQ 0
: 1682      1760  6      THEN
: 1683      1761  6      BEGIN
: 1684      1762  6
: 1685      1763  6      IF .PASS EQL 1
: 1686      1764  7      THEN
: 1687      1765  7      FFLD = FFLD[FFLD_NEXT] = GET_FFLD(.NAM, 0);
: 1688      1766  7      IF .PASS EQL 3
: 1689      1767  7      THEN
: 1690      1768  7      PUT_TEXT(
: 1691      1769  7      _LENADR_(' %0'),
: 1692      1770  7      3, CVT_BTO(.NAM));
: 1693      1771  7      ELSE
: 1694      1772  7
: 1695      1773  7
: 1696      1774  7
: 1697      1775  7      IF
: 1698      1776  7      %fi
: 1699      1777  7      EXITLOOP;
: 1700      1778  6      END;
: 1701      1779  6      END;
: 1702      1780  5      END;
: 1703      1781  4      END;
: 1704      1782  4      END;
: 1705      1783  4      IF .PASS EQL 3 THEN PUT_TEXT(LENADR_('')));
: 1706      1784  4      END;
: 1707      1785  4      IF .PASS EQL 3 THEN PUT_TEXT(LENADR_('')));
: 1708      1786  4      END;
: 1709      1787  4      IF
: 1710      1788  3      %fi
: 1711      1789  3      END;
: 1712      1790  3      ['D']: ! Data field
: 1713      1791  4      BEGIN
: 1714      1792  4      LOCAL NAM;
: 1715      1793  4      NAM = GET_DEFN(8);
: 1716      1794  4      IF .PASS EQL 3
: 1717      1795  4      THEN
: 1718      1796  5      BEGIN
: 1719      1797  5      PUT_NEST(.NESTED);
: 1720      1798  5      PUT_TEXT(LENADR_(X_DATA, '='));
```



```

: 1721      1799  5      PUT_NAME(.NAM);
: 1722      1800  5      CHECK_LENGTH(.NAM);
: 1723      1801  4      END;
: 1724      1802  4      GET_LINE();
: 1725      1803  3      END;
: 1726      1804  3
: 1727      1805  3
: 1728      1806  3      ! Note that the keys (except for user-specified forces)
: 1729      1807  3      ! are not written by this routine.
: 1730      1808  3
: 1731      1809  3      ['N']:
: 1732      1810  4      BEGIN
: 1733      1811  4      IF .PASS EQL 1
: 1734      1812  4      THEN
: 1735      1813  4      BBLOCK[ NEW_KEY(GET_DEFN(8)), KEY_ASCE ] = .ASCENDING;
: 1736      1814  4      GET_LINE();
: 1737      1815  3      END;
: 1738      1816  3      ['O']:
: 1739      1817  4      BEGIN
: 1740      1818  4      IF .PASS EQL 1
: 1741      1819  4      THEN
: 1742      1820  4      BBLOCK[ NEW_KEY(GET_DEFN(8)), KEY_ASCE ] = NOT .ASCENDING;
: 1743      1821  4      GET_LINE();
: 1744      1822  3      END;
: 1745      1823  3
: 1746      1824  3      [OTHERWISE]:
: 1747      1825  4      BEGIN
: 1748      1826  4      IF .PASS EQL 1
: 1749      1827  4      THEN
: 1750      1828  4      SOR_ERROR(SRTTRN$_INV_FIELD);
: 1751      1829  4      GET_LINE();
: 1752      1830  3      END;
: 1753      1831  3
: 1754      1832  3      TES;
: 1755      1833  3      END;
: 1756      1834  2      RETURN;
: 1757      1835  2
: 1758      1836  2
: 1759      1837  2
: 1760      1838  1      END;
```

```

                                .PSECT  SPLITS,NOWRT,NOEXE,2
4D  41  4E  28  3D  4E  4F  49  54  49  44  4E  4F  43  2F  000EC P.ABD: .ASCII  \ /CONDITION=(NAME:COND_\
                                5F  44  4E  4F  43  3A  45  000FB
                                28  3A  54  53  45  54  20  2C  00102 P.ABE: .ASCII  \ , TEST:(\
                                4F  25  20  51  45  20  0010A P.ABF: .ASCII  \ EQ %0\
                                29  29  00110 P.ABG: .ASCII  \))\
                                3D  41  54  41  44  00112 P.ABH: .ASCII  \DATA=\
```

.PSECT \$CODE\$,NOWRT,2

OFFC 00000 DO_FIELD:

SRTTRN
V04-000

H 16
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 60
(20)

PC	Op	OpC	OpD	OpI	OpJ	OpK	OpL	OpM	OpN	OpO	OpP	OpQ	OpR	OpS	OpT	OpU	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

			02	DD	000C5	8\$:	PUSHL	#2		1698
			01	DD	000C7		PUSHL	#1		
		0C	AE	DD	000C9		PUSHL	POS		
	0000V	CF	03	FB	000CC		CALLS	#3, TREE_INSERT		
		53	50	DD	000D1		MOVL	R0, NAM		
		50	64	DD	000D4		MOVL	PASS, R0		1699
		01	50	D1	000D7		CMPL	R0, #1		1703
			1A	12	000DA		BNEQ	9\$		
50	FC	A4	01	C1	000DC		ADDL3	#1, COND_NUM, R0		1706
	FC	A4	50	DD	000E1		MOVL	R0, COND_NUM		
		7E	50	DD	000E5		PUSHL	R0		
		69	01	A5	9A 000E7		MOVZBL	INP_BUF+17, -(SP)		1705
		62	02	FB	000EB		CALLS	#2, GET_FFLD		
		52	50	DD	000EE		MOVL	R0, (FFLD)		
			50	DD	000F1		MOVL	R0, FFLD		
		02	45	11	000F4		BRB	10\$		1699
			50	D1	000F6	9\$:	CMPL	R0, #2		1711
			40	12	000F9		BNEQ	10\$		
			57	DD	000FB		PUSHL	R7		1716
			08	DD	000FD		PUSHL	#8		1713
50	FC	A4	01	C1	000FF		ADDL3	#1, COND_NUM, R0		1715
	FC	A4	50	DD	00104		MOVL	R0, COND_NUM		
			50	DD	00108		PUSHL	R0		
	05FC	C6	01	FB	0010A		CALLS	#1, CVT_BT0		
			50	DD	0010F		PUSHL	R0		
			03	DD	00111		PUSHL	#3		1713
			EA	A7	9F 00113		PUSHAB	P.ABD		1714
			16	DD	00116		PUSHL	#22		1713
		6A	06	FB	00118		CALLS	#6, PUT_TEXT		
			53	DD	0011B		PUSHL	NAM		1717
	0000V	CF	01	FB	0011D		CALLS	#1, PUT_NAME		
			EA	A7	9F 00122		PUSHAB	P.ABG		1721
			02	DD	00125		PUSHL	#2		1718
		7E	65	9A 00127		MOVZBL	INP_BUF+16, -(SP)		1720	
	05FC	C6	01	FB	0012A		CALLS	#1, CVT_BT0		
			50	DD	0012F		PUSHL	R0		
			03	DD	00131		PUSHL	#3		1718
			08	A7	9F 00133		PUSHAB	P.ABF		1719
			06	DD	00136		PUSHL	#6		1718
		6A	06	FB	00138		CALLS	#6, PUT_TEXT		
	0000V	CF	00	FB	0013B	10\$:	CALLS	#0, GET_LINE		1744
			EA	A5	9F 00140		PUSHAB	INP_BUF		1746
			08	DD	00143		PUSHL	#8		
	0000V	CF	02	FB	00145		CALLS	#2, UPCASE		
		04	EA	A4	D1 0014A		CMPL	LINE_TYPE, #4		1750
			1D	12	0014E		BNEQ	12\$		
	46	8F	EA	A5	91 00150		CMPB	INP_BUF+6, #70		1751
			16	12	00155		BNEQ	12\$		
		20	EA	A5	91 00157		CMPB	INP_BUF+18, #32		1752
			10	13	0015B		BEQL	12\$		
		20	65	91 0015D		CMPB	INP_BUF+16, #32		1754	
			06	12	00160		BNEQ	11\$		
		53	EA	A5	9A 00162		MOVZBL	INP_BUF+17, NAM		1755
			11	11	00166		BRB	14\$		
		53	01	CE	00168	11\$:	MNEGL	#1, NAM		1756
			0C	11	0016B		BRB	14\$		1754
		06	EA	E9	0016D	12\$:	BLBC	ASCENDING, 13\$		1758

	53	FF	8F	9A	00171	MOVZBL	#255, NAM	1759
			02	11	00175	BRB	14\$	
			53	D4	00177	CLRL	NAM	1760
			03	18	00179	BGEQ	15\$	1762
		FF	29	31	0017B	BRW	7\$	
	01		64	D1	0017E	CMPL	PASS, #1	1767
			0D	12	00181	BNEQ	16\$	
			7E	D4	00183	CLRL	-(SP)	1769
			53	DD	00185	PUSHL	NAM	
	69		02	FB	00187	CALLS	#2, GET_FFLD	
	62		50	DD	0018A	MOVL	RO, (FFCD)	
	52		50	DD	0018D	MOVL	RO, FFLD	
		FE	9A	31	00190	BRW	1\$	1764
	44	8F	50	91	00193	CMPB	RO, #68	1790
			2D	12	00197	BNEQ	18\$	
			08	DD	00199	PUSHL	#8	1793
	6B		01	FB	0019B	CALLS	#1, GET_DEFN	
	52		50	DD	0019E	MOVL	RO, NAM	
	03		64	D1	001A1	CMPL	PASS, #3	1794
			6D	12	001A4	BNEQ	21\$	
		04	AC	DD	001A6	PUSHL	NESTED	1797
	0784	C6	01	FB	001A9	CALLS	#1, PUT_NEST	
			10	A7	9F	PUSHAB	P.ABH	1798
			05	DD	001B1	PUSHL	#5	
	6A		02	FB	001B3	CALLS	#2, PUT_TEXT	
			52	DD	001B6	PUSHL	NAM	1799
	0000V	CF	01	FB	001B8	CALLS	#1, PUT_NAME	
			52	DD	001BD	PUSHL	NAM	1800
	07A9	C6	01	FB	001BF	CALLS	#1, CHECK_LENGTH	
			4D	11	001C4	BRB	21\$	1802
	4E	8F	50	91	001C6	CMPB	RO, #78	1809
			18	12	001CA	BNEQ	19\$	
			64	D1	001CC	CMPL	PASS, #1	1811
			42	12	001CF	BNEQ	21\$	
			08	DD	001D1	PUSHL	#8	1813
	6B		01	FB	001D3	CALLS	#1, GET_DEFN	
			50	DD	001D6	PUSHL	RO	
	68		01	FB	001D8	CALLS	#1, NEW KEY	
08	A0	01	00	D8	A4	INSV	ASCENDING, #0, #1, 8(RO)	
			2F	11	001E2	BRB	21\$	1814
	4F	8F	50	91	001E4	CMPB	RO, #79	1816
			1B	12	001E8	BNEQ	20\$	
			64	D1	001EA	CMPL	PASS, #1	1818
			24	12	001ED	BNEQ	21\$	
			08	DD	001EF	PUSHL	#8	1820
	6B		01	FB	001F1	CALLS	#1, GET_DEFN	
			50	DD	001F4	PUSHL	RO	
	68		01	FB	001F6	CALLS	#1, NEW KEY	
08	A0	01	51	D8	A4	MCOML	ASCENDING, R1	
			00		51	INSV	R1, #0, #1, 8(RO)	
			0E	11	00203	BRB	21\$	1821
			64	D1	00205	CMPL	PASS, #1	1826
			09	12	00208	BNEQ	21\$	
		001C8024	8F	DD	0020A	PUSHL	#1867812	1828
	66		01	FB	00210	CALLS	#1, SOR_ERROR	
0000V	CF		00	FB	00213	CALLS	#0, GET_LINE	1829
			FE	12	31	BRW	1\$	1607

SRTTRN
V04-000

K 16
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 63
(20)

04 0021B

RET

; 1838

; Routine Size: 540 bytes, Routine Base: \$CODE\$ + 07F7

```
1762 1839 1 ROUTINE DO_RECORD: NOVALUE =
1763 1840 1
1764 1841 1 ++
1765 1842 1
1766 1843 1 FUNCTIONAL DESCRIPTION:
1767 1844 1
1768 1845 1 Process one record type corresponding to an include or an omit line.
1769 1846 1 Finds all fields associated with that include or omit.
1770 1847 1
1771 1848 1 In pass 1, define the fields that will be needed.
1772 1849 1 In pass 2, describe the condition.
1773 1850 1 In pass 3, describe the omit/include.
1774 1851 1
1775 1852 1 FORMAL PARAMETERS:
1776 1853 1
1777 1854 1 NONE
1778 1855 1
1779 1856 1 IMPLICIT INPUTS:
1780 1857 1
1781 1858 1 NONE
1782 1859 1
1783 1860 1 IMPLICIT OUTPUTS:
1784 1861 1
1785 1862 1 NONE
1786 1863 1
1787 1864 1 ROUTINE VALUE:
1788 1865 1
1789 1866 1 0 or 1
1790 1867 1
1791 1868 1 SIDE EFFECTS:
1792 1869 1
1793 1870 1 NONE
1794 1871 1 --
1795 1872 2 BEGIN
1796 1873 2 LOCAL
1797 1874 2 LEN,
1798 1875 2 UNCOND,
1799 1876 2 NESTED;
1800 1877 2
1801 1878 2 NESTED = FALSE;
1802 1879 2 UNCOND = FALSE;
1803 1880 2
1804 1881 2 ! Define a new record format
1805 1882 2 !
1806 1883 2 FMT_CURR = .FMT_CURR[FMT_NEXT];
1807 1884 2 IF FMT_CURR[BASE_] EQL 0 THEN FMT_CURR = NEW_FMT();
1808 1885 2
1809 1886 2 IF .LINE_TYPE EQL RECORD_LINE
1810 1887 2 THEN
1811 1888 2 BEGIN
1812 1889 2
1813 1890 2 IF CH$EQL(39-6, INP_BUF[6], 0, INP_BUF[6], %C' ')
1814 1891 2 THEN
1815 1892 2 UNCOND = TRUE
1816 1893 2 ELSE
1817 1894 2 NESTED = TRUE;
1818 1895 2
```



```
SELECTONE .PASS OF
SET
[1]:BEGIN
0;
END;
[2]:BEGIN
IF NOT .UNCOND
THEN
BEGIN
PUT_TEXT(
LENADR('/',X COND,'=(NAME:COND ')
3,CVT_BTO(COND_NUM = COND_NUM + 1),
LENADR('/',X TEST,':'));
DECR I FROM .FMT CORR[FMT COND]-1 TO 0 DO
PUT_TEXT(LENADR('('));
END;
END;
[3]:BEGIN
IF .INP BUF[5] EQL %C'O'
THEN PUT_TEXT(LENADR('/',X OMIT))
ELSE PUT_TEXT(LENADR('/',X INCL));
IF NOT .UNCOND
THEN
PUT_TEXT(
LENADR('=(',X COND,':COND ')
3,CVT_BTO(COND_NUM = COND_NUM + 1));
END;
TES;
END;
! While record is continued
!
WHILE .LINE_TYPE EQL RECORD_LINE DO
BEGIN
LOCAL
NAM;
! Don't process the conditions for unconditional omits/includes.
!
IF .UNCOND
THEN
BEGIN
GET_LINE();
EXITLOOP;
END;
! Convert to upper case
!
UPCASE(.INP_REC_SIZ, INP_BUF);
! Get the first field
!
NAM = GET_DEFN(8);
IF .PASS EQL 2
THEN
BEGIN
PUT_NAME(.NAM);
```

! Get the definition


```
PUT_TEXT(
  LENADR(' '),
  2, INP_BUF[16],
  LENADR(' '));
! The operator
END;

! Is there a second field or constant?
SELECTONE .INP_BUF[18] OF
  SET
  ['F']:
    BEGIN
      NAM = GET_DEFN(19);
      IF .PASS EQL 2 THEN PUT_NAME(.NAM);
    END;
  ['C']:
    BEGIN
      IF .PASS EQL 2
      THEN
        PUT_LITE(MINU(20, .INP_REC_SIZ-19), INP_BUF[19], .NAM);
    END;
  [OTHERWISE]:
    BEGIN
      IF .PASS EQL 2
      THEN
        SOR_ERROR(SRTTRN$_INV_CONST);
    END;
  TES;

SELECTONE .PASS OF
  SET
  [1]: FMT_CURR[FMT_COND] = .FMT_CURR[FMT_COND] + 1;
  [2]: PUT_TEXT(LENADR(' '));
  [3]: 0;
  TES;

GET_LINE();

IF .LINE_TYPE NEQ RECORD_LINE OR .INP_BUF[6] EQL %C' '
THEN
  EXITLOOP;
! It's not a continuation

! Convert to upper case
UPCASE(.INP_REC_SIZ, INP_BUF);

! Is it an "and" or an "or"
IF .PASS EQL 2
THEN
  BEGIN
    SELECTONE .INP_BUF[6] OF
      SET
      ['A']: PUT_TEXT(LENADR(' ',X_AND,' '));
      ['O']: PUT_TEXT(LENADR(' ',X_OR,' '));
      [OTHERWISE]: SOR_ERROR(SRTTRN$_INV_CONT);
    TES;
```



```

: 1933      2010      3      END;
: 1934      2011      3
: 1935      2012      2      END;
: 1936      2013      2
: 1937      2014      2
: 1938      2015      2      ! Issue the keys for the current record format
: 1939      2016      2      !
: 1940      2017      2      IF .PASS EQL 3
: 1941      2018      2      THEN
: 1942      2019      3      BEGIN
: 1943      2020      3      LOCAL
: 1944      2021      3      KEY: REF KEY_BLOCK;
: 1945      2022      3      KEY = FMT_CURR[FMT_KEYS] - %FIELDEXPAND(KEY_NEXT,0);
: 1946      2023      3      WHILE (KEY = .KEY[KEY_NEXT]) NEQ 0 DO
: 1947      2024      4      BEGIN
: 1948      2025      4      LOCAL
: 1949      2026      4      FLD: REF FLD_BLOCK,
: 1950      2027      4      AD;
: 1951      2028      4      PUT_NEST(.NESTED);
: 1952      2029      4      PUT_TEXT(
: 1953      2030      4      -LENADR(X_KEY,'=('));
: 1954      2031      4      FLD = .KEY[KEY_FLD];
: 1955      2032      4      IF FLD[BASE_] EQL 0
: 1956      2033      4      THEN
: 1957      2034      5      BEGIN
: 1958      2035      5      PUT_TEXT(
: 1959      2036      5      -LENADR('%0'),
: 1960      2037      5      3, CVT_BTO(.KEY[KEY_LIT]));
: 1961      2038      5      AD = .ASCENDING;
: 1962      2039      5      END
: 1963      2040      4      ELSE
: 1964      2041      5      BEGIN
: 1965      2042      5      %if k_new
: 1966      2043      5      %then
: 1967      2044      5      IF .FLD[FLD_FFLD] NEQ 0
: 1968      2045      5      THEN
: 1969      2046      5      PUT_FFLD(FLD[BASE_], .NESTED)
: 1970      2047      5      ELSE
: 1971      2048      5      %else
: 1972      2049      5      %fi
: 1973      2050      5      PUT_NAME(FLD[BASE_]);
: 1974      2051      5      AD = .KEY[KEY_ASCE];
: 1975      2052      4      END;
: 1976      2053      4      IF .AD
: 1977      2054      4      THEN PUT_TEXT(LENADR(',%CHAR(9),X_ASCE,')'))
: 1978      2055      4      ELSE PUT_TEXT(LENADR(',%CHAR(9),X_DESC,')'));
: 1979      2056      4
: 1980      2057      4      ! If keys are not being stripped, also issue this as a data field.
: 1981      2058      4      !
: 1982      2059      4      IF FLD[BASE_] NEQ 0
: 1983      2060      4      THEN
: 1984      2061      5      BEGIN
: 1985      2062      5      IF .NOSTRIP
: 1986      2063      5      THEN
: 1987      2064      6      BEGIN
: 1988      2065      6      PUT_NEST(.NESTED);
: 1989      2066      6      PUT_TEXT(LENADR(X_DATA,'='));

```



```
: 1990      L 2067 6 %if k_new
: 1991      2068 6 %then
: 1992      2069 6
: 1993      2070 6
: 1994      2071 7
: 1995      2072 7
: 1996      2073 7
: 1997      2074 7
: 1998      2075 7
: 1999      2076 6
: 2000      U 2077 6 %else
: 2001      2078 6 %fi
: 2002      2079 6
: 2003      2080 6
: 2004      2081 6
: 2005      2082 5
: 2006      2083 4
: 2007      2084 5
: 2008      2085 5
: 2009      2086 5
: 2010      2087 4
: 2011      2088 3
: 2012      2089 3
: 2013      2090 2
: 2014      2091 2
: 2015      2092 2
: 2016      2093 2
: 2017      2094 2
: 2018      2095 2
: 2019      2096 2
: 2020      2097 2
: 2021      2098 2
: 2022      2099 2
: 2023      2100 2
: 2024      2101 2
: 2025      2102 1

      IF .FLD[FLD_FFLD] NEQ 0
      THEN
      BEGIN
      PUT_TEXT(LENADR('('));
      PUT_FFLD(FLD[BASE_], .NESTED);
      PUT_TEXT(LENADR(')'));
      END
      ELSE
      BEGIN
      PUT_NAME(FLD[BASE_]);
      CHECK_LENGTH(FLD[BASE_]);
      END
      ELSE
      BEGIN
      IF NOT .ADDINCKEY THEN SOR_ERROR(SRTTRN$_ADDINCKEY);
      ADDINCKEY = TRUE;
      END;
      END;

      END;

      ! Now look for all fields defined for this record type
      !
      WHILE .LINE_TYPE EQL FIELD_LINE DO DO_FIELD(.NESTED);

      ! Close the omit, include or condition statement
      !
      IF .NESTED AND .PASS NEQ 1
      THEN
      PUT_TEXT(LENADR(')'));

      END;
```

```
                                .PSECT $SPLITS$,NOWRT,NOEXE,2
4D 41 4E 28 3D 4E 4F 49 54 49 44 4E 4F 43 2F 00117 P.ABI: .ASCII  \ /CONDITION=(NAME:COND_\
      5F 44 4E 4F 43 3A 45 00126
      3A 54 53 45 54 20 2C 0012D P.ABJ: .ASCII  \ , TEST:\
      28 00134 P.ABK: .ASCII  \ ( \
      54 49 4D 4F 2F 00135 P.ABL: .ASCII  \ /OMIT\
      45 44 55 4C 43 4E 49 2F 0013A P.ABM: .ASCII  \ /INCLUDE\
4E 4F 43 3A 4E 4F 49 54 49 44 4E 4F 43 28 3D 00142 P.ABN: .ASCII  \ =(CONDITION:COND_\
      5F 44 00151
      20 00153 P.ABO: .ASCII  \ \
      20 00154 P.ABP: .ASCII  \ \
      29 00155 P.ABQ: .ASCII  \ ) \
      20 44 4E 41 20 00156 P.ABR: .ASCII  \ AND \
      20 20 52 4F 20 0015B P.ABS: .ASCII  \ OR \
      28 3D 59 45 4B 00160 P.ABT: .ASCII  \ KEY=(\
      4F 25 00165 P.ABU: .ASCII  \ X \
      29 47 4E 49 44 4E 45 43 53 41 09 2C 00167 P.ABV: .ASCII  \ , \<9>\ASCENDING)\
```


SRTTRN
V04-000

E 1
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 69
(21)

```

29 47 4E 49 44 4E 45 43 53 45 44 09 2C 00173 P.ABW: .ASCII \,\<9>\DESCENDING)\
3D 41 54 41 44 00180 P.ABX: .ASCII \DATA=\
28 00185 P.ABY: .ASCII \(\
29 00186 P.ABZ: .ASCII \)\
29 00187 P.ACA: .ASCII \)\

```

```
.PSECT $CODE$,NOWRT,2
```

OFFC 00000 DO_RECORD:

PC	PC+4	PC+8	PC+12	PC+16	PC+20	PC+24	PC+28	PC+32	PC+36	PC+40	PC+44	PC+48	PC+52	PC+56	PC+60	PC+64	PC+68	PC+72	PC+76	PC+80	PC+84	PC+88	PC+92	PC+96	PC+100	PC+104	PC+108	PC+112	PC+116	PC+120	PC+124	PC+128	PC+132	PC+136	PC+140	PC+144	PC+148	PC+152	PC+156	PC+160	PC+164	PC+168	PC+172	PC+176	PC+180	PC+184	PC+188	PC+192	PC+196	PC+200	PC+204	PC+208	PC+212	PC+216	PC+220	PC+224	PC+228	PC+232	PC+236	PC+240	PC+244	PC+248	PC+252	PC+256	PC+260	PC+264	PC+268	PC+272	PC+276	PC+280	PC+284	PC+288	PC+292	PC+296	PC+300	PC+304	PC+308	PC+312	PC+316	PC+320	PC+324	PC+328	PC+332	PC+336	PC+340	PC+344	PC+348	PC+352	PC+356	PC+360	PC+364	PC+368	PC+372	PC+376	PC+380	PC+384	PC+388	PC+392	PC+396	PC+400	PC+404	PC+408	PC+412	PC+416	PC+420	PC+424	PC+428	PC+432	PC+436	PC+440	PC+444	PC+448	PC+452	PC+456	PC+460	PC+464	PC+468	PC+472	PC+476	PC+480	PC+484	PC+488	PC+492	PC+496	PC+500	PC+504	PC+508	PC+512	PC+516	PC+520	PC+524	PC+528	PC+532	PC+536	PC+540	PC+544	PC+548	PC+552	PC+556	PC+560	PC+564	PC+568	PC+572	PC+576	PC+580	PC+584	PC+588	PC+592	PC+596	PC+600	PC+604	PC+608	PC+612	PC+616	PC+620	PC+624	PC+628	PC+632	PC+636	PC+640	PC+644	PC+648	PC+652	PC+656	PC+660	PC+664	PC+668	PC+672	PC+676	PC+680	PC+684	PC+688	PC+692	PC+696	PC+700	PC+704	PC+708	PC+712	PC+716	PC+720	PC+724	PC+728	PC+732	PC+736	PC+740	PC+744	PC+748	PC+752	PC+756	PC+760	PC+764	PC+768	PC+772	PC+776	PC+780	PC+784	PC+788	PC+792	PC+796	PC+800	PC+804	PC+808	PC+812	PC+816	PC+820	PC+824	PC+828	PC+832	PC+836	PC+840	PC+844	PC+848	PC+852	PC+856	PC+860	PC+864	PC+868	PC+872	PC+876	PC+880	PC+884	PC+888	PC+892	PC+896	PC+900	PC+904	PC+908	PC+912	PC+916	PC+920	PC+924	PC+928	PC+932	PC+936	PC+940	PC+944	PC+948	PC+952	PC+956	PC+960	PC+964	PC+968	PC+972	PC+976	PC+980	PC+984	PC+988	PC+992	PC+996	PC+1000	PC+1004	PC+1008	PC+1012	PC+1016	PC+1020	PC+1024	PC+1028	PC+1032	PC+1036	PC+1040	PC+1044	PC+1048	PC+1052	PC+1056	PC+1060	PC+1064	PC+1068	PC+1072	PC+1076	PC+1080	PC+1084	PC+1088	PC+1092	PC+1096	PC+1100	PC+1104	PC+1108	PC+1112	PC+1116	PC+1120	PC+1124	PC+1128	PC+1132	PC+1136	PC+1140	PC+1144	PC+1148	PC+1152	PC+1156	PC+1160	PC+1164	PC+1168	PC+1172	PC+1176	PC+1180	PC+1184	PC+1188	PC+1192	PC+1196	PC+1200	PC+1204	PC+1208	PC+1212	PC+1216	PC+1220	PC+1224	PC+1228	PC+1232	PC+1236	PC+1240	PC+1244	PC+1248	PC+1252	PC+1256	PC+1260	PC+1264	PC+1268	PC+1272	PC+1276	PC+1280	PC+1284	PC+1288	PC+1292	PC+1296	PC+1300	PC+1304	PC+1308	PC+1312	PC+1316	PC+1320	PC+1324	PC+1328	PC+1332	PC+1336	PC+1340	PC+1344	PC+1348	PC+1352	PC+1356	PC+1360	PC+1364	PC+1368	PC+1372	PC+1376	PC+1380	PC+1384	PC+1388	PC+1392	PC+1396	PC+1400	PC+1404	PC+1408	PC+1412	PC+1416	PC+1420	PC+1424	PC+1428	PC+1432	PC+1436	PC+1440	PC+1444	PC+1448	PC+1452
----	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------

50	4F	8F	FF	33	12	00090	7\$:	BNEQ	10\$		
				A9	91	00092		CMPB	INP_BUF+5, #79		1914
			08	07	12	00097		BNEQ	8\$		
				A7	9F	00099		PUSHAB	P.ABL		1915
				05	DD	0009C		PUSHL	#5		
				05	11	0009E		BRB	9\$		
			0D	A7	9F	000A0	8\$:	PUSHAB	P.ABM		1916
				08	DD	000A3		PUSHL	#8		
		68		02	FB	000A5	9\$:	CALLS	#2, PUT_TEXT		
		1A		55	E8	000A8		BLBS	UNCOND, -10\$		1917
	FC	A6		01	C1	000AB		ADDL3	#1, COND_NUM, R0		1921
	FC	A6		50	DD	000B0		MOVL	R0, COND_NUM		
				50	DD	000B4		PUSHL	R0		
		6A		01	FB	000B6		CALLS	#1, CVT_BTO		
				50	DD	000B9		PUSHL	R0		
				03	DD	000BB		PUSHL	#3		1919
			15	A7	9F	000BD		PUSHAB	P.ABN		1920
				11	DD	000C0		PUSHL	#17		1919
		68		04	FB	000C2		CALLS	#4, PUT_TEXT		
		03	EC	A6	D1	000C5	10\$:	CMPL	LINE_TYPE, #3		1928
				08	12	000C9		BNEQ	11\$		
		08		55	E9	000CB		BLBC	UNCOND, 12\$		1935
0000V		CF		00	FB	000CE		CALLS	#0, GET_LINE		1938
				00F4	31	000D3	11\$:	BRW	24\$		1937
			FA	A9	9F	000D6	12\$:	PUSHAB	INP_BUF		1944
			0186	C9	DD	000D9		PUSHL	INP_REC_SIZ		
0000V		CF		02	FB	000DD		CALLS	#2, UPCASE		
				08	DD	000E2		PUSHL	#8		1948
0000V		CF		01	FB	000E4		CALLS	#1, GET_DEFN		
		52		50	DD	000E9		MOVL	R0, NAM		
		02		66	D1	000EC		CMPL	PASS, #2		1949
				17	12	000EF		BNEQ	13\$		
				52	DD	000F1		PUSHL	NAM		1952
		6B		01	FB	000F3		CALLS	#1, PUT_NAME		
			27	A7	9F	000F6		PUSHAB	P.ABP		1956
				01	DD	000F9		PUSHL	#1		1955
			0A	A9	9F	000FB		PUSHAB	INP_BUF+16		
				02	DD	000FE		PUSHL	#2		
			26	A7	9F	00100		PUSHAB	P.ABO		1954
				01	DD	00103		PUSHL	#1		1955
		68		06	FB	00105		CALLS	#6, PUT_TEXT		
		50	OC	A9	9A	00108	13\$:	MOVZBL	INP_BUF+18, R0		1961
46		8F		50	91	0010C		CMPB	R0, #70		1963
				16	12	00110		BNEQ	14\$		
				13	DD	00112		PUSHL	#19		1965
0000V		CF		01	FB	00114		CALLS	#1, GET_DEFN		
		52		50	DD	00119		MOVL	R0, NAM		
		02		66	D1	0011C		CMPL	PASS, #2		1966
				3E	12	0011F		BNEQ	17\$		
				52	DD	00121		PUSHL	NAM		
		6B		01	FB	00123		CALLS	#1, PUT_NAME		
				37	11	00126		BRB	17\$		1961
43		8F		50	91	00128	14\$:	CMPB	R0, #67		1968
				21	12	0012C		BNEQ	16\$		
		02		66	D1	0012E		CMPL	PASS, #2		1970
				2C	12	00131		BNEQ	17\$		
				52	DD	00133		PUSHL	NAM		1972

50	0186	C9	OD	A9	9F	00135	PUSHAB	INP_BUF+19	:	
				13	C3	00138	SUBL3	#19, INP_REC_SIZ, R0	:	
		14		50	DD	0013E	PUSHL	R0	:	
				6E	D1	00140	CMPL	(SP), #20	:	
		6E		03	1B	00143	BLEQU	15\$	:	
	0000V	CF		14	D0	00145	MOVL	#20, (SP)	:	
				03	FB	00148	CALLS	#3, PUT_LITE	:	
		02		10	11	0014D	BRB	17\$	:	1961
				66	D1	0014F	CMPL	PASS, #2	:	1976
				0B	12	00152	BNEQ	17\$	:	
	FA04	CA	001C800C	8F	DD	00154	PUSHL	#1867788	:	1978
		50		01	FB	0015A	CALLS	#1, SOR_ERROR	:	
		01		66	D0	0015F	MOVL	PASS, R0	:	1982
				50	D1	00162	CMPL	R0, #1	:	1984
				09	12	00165	BNEQ	18\$	:	
		50	E8	A6	D0	00167	MOVL	FMT_CURR, R0	:	
			10	A0	B6	0016B	INCW	16(R0)	:	
				0D	11	0016E	BRB	19\$	:	
		02		50	D1	00170	CMPL	R0, #2	:	1985
				08	12	00173	BNEQ	19\$	:	
			28	A7	9F	00175	PUSHAB	P.ABQ	:	
				01	DD	00178	PUSHL	#1	:	
	0000V	68		02	FB	0017A	CALLS	#2, PUT_TEXT	:	
		CF		00	FB	0017D	CALLS	#0, GET_LINE	:	1989
		03	EC	A6	D1	00182	CMPL	LINE_TYPE, #3	:	1991
				42	12	00186	BNEQ	24\$	:	
		20		69	91	00188	CMPB	INP_BUF+6, #32	:	
				3D	13	0018B	BEQL	24\$	:	
			FA	A9	9F	0018D	PUSHAB	INP_BUF	:	1997
			0186	C9	DD	00190	PUSHL	INP_REC_SIZ	:	
	0000V	CF		02	FB	00194	CALLS	#2, UPCASE	:	
		02		66	D1	00199	CMPL	PASS, #2	:	2001
				29	12	0019C	BNEQ	23\$	:	
		50		69	9A	0019E	MOVZBL	INP_BUF+6, R0	:	2004
	41	8F		50	91	001A1	CMPB	R0, #65	:	2006
				05	12	001A5	BNEQ	20\$	:	
			29	A7	9F	001A7	PUSHAB	P.ABR	:	
				09	11	001AA	BRB	21\$	:	
	4F	8F		50	91	001AC	CMPB	R0, #79	:	2007
				0A	12	001B0	BNEQ	22\$	:	
			2E	A7	9F	001B2	PUSHAB	P.ABS	:	
				05	DD	001B5	PUSHL	#5	:	
		68		02	FB	001B7	CALLS	#2, PUT_TEXT	:	
				0B	11	001BA	BRB	23\$	:	
			001C8014	8F	DD	001BC	PUSHL	#1867796	:	2008
	FA04	CA		01	FB	001C2	CALLS	#1, SOR_ERROR	:	
				FEFB	31	001C7	BRW	10\$	:	2001
		03		66	D1	001CA	CMPL	PASS, #3	:	2017
				03	13	001CD	BEQL	26\$	:	
			00BC	31	001CF	25\$:	BRW	39\$	:	
53		E8		04	C1	001D2	ADDL3	#4, FMT_CURR, KEY	:	2022
		53		63	D0	001D7	MOVL	(KEY), KEY	:	2023
				F3	13	001DA	BEQL	25\$	:	
				54	DD	001DC	PUSHL	NESTED	:	2028
	0188	CA		01	FB	001DE	CALLS	#1, PUT_NEST	:	
			33	A7	9F	001E3	PUSHAB	P.ABT	:	2030
				05	DD	001E6	PUSHL	#5	:	2029

68		02	FB	001E8	CALLS	#2, PUT_TEXT	:	2031
52	04	A3	DO	001EB	MOVL	4(KEY), -FLD	:	2032
7E		19	12	001EF	BNEQ	28\$	:	2037
6A	08	A3	9A	001F1	MOVZBL	8(KEY), -(SP)	:	
		01	FB	001F5	CALLS	#1, CVT_BTO	:	
		50	DD	001F8	PUSHL	R0	:	
		03	DD	001FA	PUSHL	#3	:	2035
	38	A7	9F	001FC	PUSHAB	P.ABU	:	2036
		02	DD	001FF	PUSHL	#2	:	2035
68		04	FB	00201	CALLS	#4, PUT_TEXT	:	
50	D8	A6	DO	00204	MOVL	ASCENDING, AD	:	2038
		19	11	00208	BRB	31\$	:	2032
	0C	A2	D5	0020A	TSTL	12(FLD)	:	2044
		09	13	0020D	BEQL	29\$	:	
		14	BB	0020F	PUSHR	#^M<R2,R4>	:	2046
0000V	CF	02	FB	00211	CALLS	#2, PUT_FFLD	:	
		05	11	00216	BRB	30\$	:	
		52	DD	00218	PUSHL	FLD	:	2050
68		01	FB	0021A	CALLS	#1, PUT_NAME	:	
50	08	00	EF	0021D	EXTZV	#0, #1, -8(KEY), AD	:	2051
A3		50	E9	00223	BLBC	AD, 32\$	:	2053
		07	9F	00226	PUSHAB	P.ABV	:	2054
	3A	0C	DD	00229	PUSHL	#12	:	
		05	11	0022B	BRB	33\$	:	
	46	A7	9F	0022D	PUSHAB	P.ABW	:	2055
		0D	DD	00230	PUSHL	#13	:	
68		02	FB	00232	CALLS	#2, PUT_TEXT	:	
		52	D5	00235	TSTL	FLD	:	2059
		3F	13	00237	BEQL	36\$	:	
9A	0C	A6	E9	00239	BLBC	NOSTRIP, 27\$	:	2062
		54	DD	0023D	PUSHL	NESTED	:	2065
0188	CA	01	FB	0023F	CALLS	#1, PUT_NEST	:	
	53	A7	9F	00244	PUSHAB	P.ABX	:	2066
		05	DD	00247	PUSHL	#5	:	
68		02	FB	00249	CALLS	#2, PUT_TEXT	:	
	0C	A2	D5	0024C	TSTL	12(FLD)	:	2069
		19	13	0024F	BEQL	34\$	:	
	58	A7	9F	00251	PUSHAB	P.ABY	:	2072
		01	DD	00254	PUSHL	#1	:	
68		02	FB	00256	CALLS	#2, PUT_TEXT	:	
		14	BB	00259	PUSHR	#^M<R2,R4>	:	2073
0000V	CF	02	FB	0025B	CALLS	#2, PUT_FFLD	:	
	59	A7	9F	00260	PUSHAB	P.ABZ	:	2074
		01	DD	00263	PUSHL	#1	:	
68		02	FB	00265	CALLS	#2, PUT_TEXT	:	
		05	11	00268	BRB	35\$	:	2069
		52	DD	0026A	PUSHL	FLD	:	2079
68		01	FB	0026C	CALLS	#1, PUT_NAME	:	
		52	DD	0026F	PUSHL	FLD	:	2080
01AD	CA	01	FB	00271	CALLS	#1, CHECK_LENGTH	:	
		13	11	00276	BRB	38\$	:	2061
08	F8	A6	E8	00278	BLBS	ADDINCKEY, 37\$	:	2085
	001C808B	8F	DD	0027C	PUSHL	#1867915	:	
FA04	CA	01	FB	00282	CALLS	#1, SOR_ERROR	:	
F8	A6	01	DO	00287	MOVL	#1, ADDINCKEY	:	2086
		FF49	31	0028B	BRW	27\$	:	2023
04	EC	A6	D1	0028E	CMPL	LINE_TYPE, #4	:	2094

1 1
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

Page 73
(21)

2098
2100
2102

; Routine Size: 686 bytes, Routine Base: \$CODE\$ + 0A13

Page 74
(22)

```
.EXTRN  LIB$GET_VM
```

PC	OP	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9	OP10	OP11	OP12	OP13	OP14	OP15	OP16	OP17	OP18	OP19	OP20	OP21	OP22	OP23	OP24	OP25	OP26	OP27	OP28	OP29	OP30	OP31	OP32	OP33	OP34	OP35	OP36	OP37	OP38	OP39	OP40	OP41	OP42	OP43	OP44	OP45	OP46	OP47	OP48	OP49	OP50	OP51	OP52	OP53	OP54	OP55	OP56	OP57	OP58	OP59	OP60	OP61	OP62	OP63	OP64	OP65	OP66	OP67	OP68	OP69	OP70	OP71	OP72	OP73	OP74	OP75	OP76	OP77	OP78	OP79	OP80	OP81	OP82	OP83	OP84	OP85	OP86	OP87	OP88	OP89	OP90	OP91	OP92	OP93	OP94	OP95	OP96	OP97	OP98	OP99	OP100	OP101	OP102	OP103	OP104	OP105	OP106	OP107	OP108	OP109	OP110	OP111	OP112	OP113	OP114	OP115	OP116	OP117	OP118	OP119	OP120	OP121	OP122	OP123	OP124	OP125	OP126	OP127	OP128	OP129	OP130	OP131	OP132	OP133	OP134	OP135	OP136	OP137	OP138	OP139	OP140	OP141	OP142	OP143	OP144	OP145	OP146	OP147	OP148	OP149	OP150	OP151	OP152	OP153	OP154	OP155	OP156	OP157	OP158	OP159	OP160	OP161	OP162	OP163	OP164	OP165	OP166	OP167	OP168	OP169	OP170	OP171	OP172	OP173	OP174	OP175	OP176	OP177	OP178	OP179	OP180	OP181	OP182	OP183	OP184	OP185	OP186	OP187	OP188	OP189	OP190	OP191	OP192	OP193	OP194	OP195	OP196	OP197	OP198	OP199	OP200	OP201	OP202	OP203	OP204	OP205	OP206	OP207	OP208	OP209	OP210	OP211	OP212	OP213	OP214	OP215	OP216	OP217	OP218	OP219	OP220	OP221	OP222	OP223	OP224	OP225	OP226	OP227	OP228	OP229	OP230	OP231	OP232	OP233	OP234	OP235	OP236	OP237	OP238	OP239	OP240	OP241	OP242	OP243	OP244	OP245	OP246	OP247	OP248	OP249	OP250	OP251	OP252	OP253	OP254	OP255	OP256	OP257	OP258	OP259	OP260	OP261	OP262	OP263	OP264	OP265	OP266	OP267	OP268	OP269	OP270	OP271	OP272	OP273	OP274	OP275	OP276	OP277	OP278	OP279	OP280	OP281	OP282	OP283	OP284	OP285	OP286	OP287	OP288	OP289	OP290	OP291	OP292	OP293	OP294	OP295	OP296	OP297	OP298	OP299	OP300	OP301	OP302	OP303	OP304	OP305	OP306	OP307	OP308	OP309	OP310	OP311	OP312	OP313	OP314	OP315	OP316	OP317	OP318	OP319	OP320	OP321	OP322	OP323	OP324	OP325	OP326	OP327	OP328	OP329	OP330	OP331	OP332	OP333	OP334	OP335	OP336	OP337	OP338	OP339	OP340	OP341	OP342	OP343	OP344	OP345	OP346	OP347	OP348	OP349	OP350	OP351	OP352	OP353	OP354	OP355	OP356	OP357	OP358	OP359	OP360	OP361	OP362	OP363	OP364	OP365	OP366	OP367	OP368	OP369	OP370	OP371	OP372	OP373	OP374	OP375	OP376	OP377	OP378	OP379	OP380	OP381	OP382	OP383	OP384	OP385	OP386	OP387	OP388	OP389	OP390	OP391	OP392	OP393	OP394	OP395	OP396	OP397	OP398	OP399	OP400	OP401	OP402	OP403	OP404	OP405	OP406	OP407	OP408	OP409	OP410	OP411	OP412	OP413	OP414	OP415	OP416	OP417	OP418	OP419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

; Routine Size: 41 bytes, Routine Base: \$CODES + 0CC1


```
: 2056      2131 1 ROUTINE NEW_FMT =  
: 2057      2132 2   BEGIN  
: 2058      2133 2   LOCAL  
: 2059      2134 2       FMTADR: REF VECTOR[1],  
: 2060      2135 2       FMT:   REF FMT_BLOCK;  
: 2061      2136 2       FMTADR = FMT_ROOT;  
: 2062      2137 2       WHILE (FMT = .FMTADR[0]) NEQ 0 DO FMTADR = FMT[FMT_NEXT];  
: 2063      2138 2       GET_VM(FMT_K SIZE, FMTADR[0]);  
: 2064      2139 2       FMT = .FMTADR[0];  
: 2065      2140 2       RETURN FMT[BASE_];  
: 2066      2141 1   END;
```

			000C 00000	NEW_FMT: .WORD	Save R2,R3	: 2131
	53	0000'	CF 9E 00002	MOVAB	FMT_ROOT, FMTADR	: 2136
	52		63 D0 00007	1\$: MOVL	(FMTADR), FMT	: 2137
			05 13 0000A	BEQL	2\$	: 2138
	53		52 D0 0000C	MOVL	FMT, FMTADR	: 2139
			F6 11 0000F	BRB	1\$	: 2140
			53 DD 00011	2\$: PUSHL	FMTADR	: 2141
			12 DD 00013	PUSHL	#18	
BE	AF		02 FB 00015	CALLS	#2, GET_VM	
	52		63 D0 00019	MOVL	(FMTADR), FMT	
	50		52 D0 0001C	MOVL	FMT, R0	
			04 0001F	RET		

; Routine Size: 32 bytes, Routine Base: \$CODE\$ + 0CEA

```

: 2068      2142 1 ROUTINE NEW_KEY(FLD: REF FLD_BLOCK, FMTX: REF FMT_BLOCK): CALL =
: 2069      2143 2 BEGIN
: 2070      2144 2
: 2071      2145 2 | Insert FLD in the key list for the current format, just before the
: 2072      2146 2 | current key definition. Leave the current key alone, and return the
: 2073      2147 2 | address of the new key definition.
: 2074      2148 2 |
: 2075      2149 2 BUILTIN
: 2076      2150 2 NULLPARAMETER;
: 2077      2151 2 LOCAL
: 2078      2152 2 KEYADR: REF VECTOR[1],
: 2079      2153 2 KEY: REF KEY_BLOCK,
: 2080      2154 2 FMT: REF FMT_BLOCK;
: 2081      2155 2 IF NOT NULLPARAMETER(2)
: 2082      2156 2 THEN FMT = FMTX[BASE_];
: 2083      2157 2 ELSE FMT = FMT_CURR[BASE_];
: 2084      2158 2 KEYADR = FMT[FMT_KEYS];
: 2085      2159 2 WHILE (KEY = .KEYADR[0]) NEQ .FMT[FMT_CKEY] DO KEYADR = KEY[KEY_NEXT];
: 2086      2160 2 GET VM(KEY_K_SIZE, KEYADR[0]);
: 2087      2161 2 BBLOCK(.KEYADR[0], KEY_NEXT) = KEY[BASE_];
: 2088      2162 2 KEY = .KEYADR[0];
: 2089      2163 2 KEY[KEY_FLD] = FLD[BASE_];
: 2090      2164 2 RETURN KEY[BASE_];
: 2091      2165 1 END;

```

			000C 00000	NEW_KEY: .WORD	Save R2,R3	
	02	6C	91 00002	CMPB	(AP), #2	: 2142
		0B	1F 00005	BLSSU	1\$	: 2155
		08	AC D5 00007	TSTL	8(AP)	
		06	13 0000A	BEQL	1\$	
	50	08	AC D0 0000C	MOVL	FMTX, FMT	: 2156
		05	11 00010	BRB	2\$	
	50	0000'	CF D0 00012	1\$: MOVL	FMT_CURR, FMT	: 2157
	52	04	A0 9E 00017	2\$: MOVAB	4(R0), KEYADR	: 2158
	53		62 D0 0001B	3\$: MOVL	(KEYADR), KEY	: 2159
08	A0		53 D1 0001E	CMPL	KEY, 8(FMT)	
			05 13 00022	BEQL	4\$	
	52		53 D0 00024	MOVL	KEY, KEYADR	
			F2 11 00027	BRB	3\$	
			52 DD 00029	4\$: PUSHL	KEYADR	: 2160
			09 DD 0002B	PUSHL	#9	
86	AF		02 FB 0002D	CALLS	#2, GET VM	
00	B2		53 D0 00031	MOVL	KEY, 80(KEYADR)	: 2161
	53		62 D0 00035	MOVL	(KEYADR), KEY	: 2162
04	A3	04	AC D0 00038	MOVL	FLD, 4(KEY)	: 2163
	50		53 D0 0003D	MOVL	KEY, R0	: 2164
			04 00040	RET		: 2165

; Routine Size: 65 bytes, Routine Base: \$CODE\$ + 0D0A


```
2093 2166 1 ROUTINE TREE_INSERT(  
2094 2167 1     POS,  
2095 2168 1     SIZ,  
2096 2169 1     DTY) =  
2097 2170 2 BEGIN  
2098 2171 2 LOCAL  
2099 2172 2     PTRADR: REF VECTOR[1],  
2100 2173 2     PTR: REF FLD_BLOCK;  
2101 2174 2  
2102 2175 2     PTRADR = FLD_ROOT;  
2103 2176 2     WHILE (PTR = .PTRADR[0]) NEQ 0 DO  
2104 2177 3 BEGIN  
2105 2178 3     IF .POS LSS .PTR[FLD_POS] THEN PTRADR = PTR[FLD_LSON]  
2106 2179 3     ELIF .POS GTR .PTR[FLD_POS] THEN PTRADR = PTR[FLD_RSON]  
2107 2180 3     ELIF .SIZ LSS .PTR[FLD_SIZ] THEN PTRADR = PTR[FLD_LSON]  
2108 2181 3     ELIF .SIZ GTR .PTR[FLD_SIZ] THEN PTRADR = PTR[FLD_RSON]  
2109 2182 3     ELIF .DTY LSS .PTR[FLD_DTY] THEN PTRADR = PTR[FLD_LSON]  
2110 2183 3     ELIF .DTY GTR .PTR[FLD_DTY] THEN PTRADR = PTR[FLD_RSON]  
2111 2184 3     ELSE  
2112 2185 3         RETURN PTR[BASE_];  
2113 2186 2     END;  
2114 2187 2  
2115 2188 2     GET_VM(FLD_K_SIZE, PTRADR[0]);  
2116 2189 2     PTR = .PTRADR[0];  
2117 2190 2     PTR[FLD_LSON] = 0;  
2118 2191 2     PTR[FLD_RSON] = 0;  
2119 2192 2     PTR[FLD_POS] = .POS;  
2120 2193 2     PTR[FLD_SIZ] = .SIZ;  
2121 2194 2     PTR[FLD_DTY] = .DTY;  
2122 2195 2  
2123 2196 2     RETURN PTR[BASE_];  
2124 2197 2  
2125 2198 1 END;
```

				000C 00000 TREE_INSERT:							
				53	0000'	CF	9E 00002	.WORD	Save R2,R3		2166
				52		63	DO 00007 1\$:	MOVAB	FLD_ROOT, PTRADR		2175
						41	13 0000A	MOVL	(PTRADR), PTR		2176
						00	ED 0000C	BEQL	5\$		
04	AC	08	A2	10		24	14 00013	CMPZV	#0, #16, 8(PTR), POS		2178
						00	ED 00015	BGTR	2\$		
04	AC	08	A2	10		29	19 0001C	CMPZV	#0, #16, 8(PTR), POS		2179
						00	ED 0001E	BLSS	4\$		
08	AC	0A	A2	10		12	14 00025	CMPZV	#0, #16, 10(PTR), SIZ		2180
						00	ED 00027	BGTR	2\$		
08	AC	0A	A2	10		17	19 0002E	CMPZV	#0, #16, 10(PTR), SIZ		2181
						00	ED 00030	BLSS	4\$		
0C	AC	10	A2	08		05	15 00037	CMPZV	#0, #8, 16(PTR), DTY		2182
						53	52 DO 00039 2\$:	BLEQ	3\$		
							C9 11 0003C	MOVL	PTR, PTRADR		
0C	AC	10	A2	08		00	ED 0003E 3\$:	BRB	1\$		
						23	18 00045	CMPZV	#0, #8, 16(PTR), DTY		2183
								BGEQ	6\$		

SRTRN
V04-000

N 1
16-Sep-1984 01:11:52 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:54 [SORT32.SRC]SRTRN.B32;1

Page 78
(25)

	53	04	A2	9E	00047	4\$:	MOVAB	4(R2), PTRADR	:	
			BA	11	0004B		BRB	1\$	:	
			53	DD	0004D	5\$:	PUSHL	PTRADR	:	2188
			11	DD	0004F		PUSHL	#17	:	
FF20	CF		02	FB	00051		CALLS	#2, GET_VM	:	
	52		63	D0	00056		MOVL	(PTRADR), PTR	:	2189
			62	7C	00059		CLRQ	(PTR)	:	2190
08	A2	04	AC	B0	0005B		MOVW	POS, 8(PTR)	:	2192
0A	A2	08	AC	B0	00060		MOVW	SIZ, 10(PTR)	:	2193
10	A2	0C	AC	90	00065		MOVB	DTY, 16(PTR)	:	2194
	50		52	D0	0006A	6\$:	MOVL	PTR, R0	:	2196
			04	0006D			RET		:	2198

; Routine Size: 110 bytes, Routine Base: \$CODE\$ + 0D4B


```
2127 1 ROUTINE MINCOMPAT(  
2128 1 K1: REF KEY_BLOCK,  
2129 1 K2: REF KEY_BLOCK,  
2130 1 RESLEN: REF VECTOR[1,WORD]) =  
2131 1 ++  
2132 1  
2133 1 Test two key descriptions for compatibility.  
2134 1  
2135 1 If the keys are compatible, return true, otherwise false.  
2136 1 If the keys are compatible, store their compatibility length in reslen.  
2137 1 This is used to split keys into several pieces, so that string keys will  
2138 1 be compared with equal size (as the old spec file did it), instead of padding  
2139 1 (as the new spec file will do it).  
2140 1  
2141 1 If (sometime) it's decided that keys of different datatypes are compatible  
2142 1 (for example, packed and decimal), reslen should be expressed in terms of k2.  
2143 1  
2144 1 --  
2145 2 BEGIN  
2146 2 LOCAL  
2147 2 F1: REF FLD_BLOCK,  
2148 2 F2: REF FLD_BLOCK;  
2149 2 IF .K1[KEY_ASCE] NEQ .K2[KEY_ASCE] THEN RETURN FALSE;  
2150 2 F1 = .K1[KEY_FLD]; ! ??? Might this be zero?  
2151 2 F2 = .K2[KEY_FLD]; ! ??? Might this be zero?  
2152 2 IF .F1[FLD_DTY] NEQ .F2[FLD_DTY] THEN RETURN FALSE;  
2153 2 SELECTONE .F1[FLD_DTY] OF  
2154 2 SET  
2155 2 [DSC$K_DTYPE_I]: 0;  
2156 2 [OTHERWISE]: IF .F1[FLD_SIZE] NEQ .F2[FLD_SIZE] THEN RETURN FALSE;  
2157 2 TES;  
2158 2 RESLEN[0] = MINU(.F1[FLD_SIZE], .F2[FLD_SIZE]);  
2159 2 RETURN TRUE;  
2160 2 END;
```

0004 00000 MINCOMPAT:									
						.WORD	Save R2		2199
		50	04	AC	7D	00002	MOVQ	K1, R0	2221
52	08	A1	08	A0	8D	00006	XORB3	8(R0), 8(R1), R2	
		34		52	E8	0000C	BLBS	R2, 3\$	
		50	04	A0	D0	0000F	MOVL	4(R0), F1	2222
		51	04	A1	D0	00013	MOVL	4(R1), F2	2223
		52	10	A0	9A	00017	MOVZBL	16(F1), R2	2224
		52	10	A1	91	0001B	CMPB	16(F2), R2	
				22	12	0001F	BNEQ	3\$	
		0E		52	91	00021	CMPB	R2, #14	2227
				07	13	00024	BEQL	1\$	
	0A	A1	0A	A0	B1	00026	CMPW	10(F1), 10(F2)	2228
				16	12	0002B	BNEQ	3\$	
		50	0A	A0	3C	0002D	MOVZWL	10(F1), R0	2230
		50	0A	A1	B1	00031	CMPW	10(F2), R0	
				04	1E	00035	BGEQU	2\$	
		50	0A	A1	3C	00037	MOVZWL	10(F2), R0	

SRTTRN
V04-000

C 2
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 80
(26)

OC BC
50

50	B0	0003B	2\$:	MOVW	R0, @RESLEN
01	D0	0003F		MOVL	#1, R0
	04	00042		RET	
50	D4	00043	3\$:	CLRL	R0
	04	00045		RET	

: 2231
: 2232
:

; Routine Size: 70 bytes, Routine Base: \$CODE\$ + 0DB9


```

: 2162      2233 1 ROUTINE DEF_FIELDS(PTR: REF FLD_BLOCK): NOVALUE =
: 2163      2234 2 BEGIN
: 2164      2235 2 IF PTR[BASE_] EQL 0 THEN RETURN;
: 2165      2236 2 DEF_FIELDS(.PTR[FLD_LSON]);
: 2166      2237 2 PUT_DEFN(PTR[BASE_]);
: 2167      2238 2 DEF_FIELDS(.PTR[FLD_RSON]);
: 2168      2239 1 END;

```

```

                                0004 00000 DEF_FIELDS:
                                .WORD
                                Save R2
                                52      04  AC  D0 00002      MOVL  PTR, R2
                                14  13 00006      BEQL  1$
                                62  DD 00008      PUSHL (R2)
                                F2  AF      01  FB 0000A      CALLS #1, DEF_FIELDS
                                0000V CF      52  DD 0000E      PUSHL R2
                                E4  AF      01  FB 00010      CALLS #1, PUT_DEFN
                                04      A2  DD 00015      PUSHL 4(R2)
                                01  FB 00018      CALLS #1, DEF_FIELDS
                                04 0001C 1$:      RET

```

```

: 2233
: 2235
: 2236
: 2237
: 2238
: 2239

```

; Routine Size: 29 bytes, Routine Base: \$CODE\$ + 0DFF


```
: 2170. 2240 1 ROUTINE DEF_SUBFIELDS: NOVALUE =
: 2171. 2241 2 BEGIN
: 2172. 2242 2
: 2173. 2243 2 | Define any subfields that may be required due to key incompatibilities
: 2174. 2244 2 | in multiple record formats.
: 2175. 2245 2
: 2176. 2246 2 LOCAL
: 2177. 2247 2 FMT: REF FMT_BLOCK;
: 2178. 2248 2
: 2179. 2249 2 | Initialize the current key being worked on for each format.
: 2180. 2250 2
: 2181. 2251 2 FMT = FMT_ROOT - %FIELDEXPAND(FMT_NEXT,0);
: 2182. 2252 2 WHILE (FMT = .FMT[FMT_NEXT]) NEQ 0 DO FMT[FMT_CKEY] = .FMT[FMT_KEYS];
: 2183. 2253 2
: 2184. 2254 2 | While there are more keys to be processed
: 2185. 2255 2
: 2186. 2256 2 WHILE TRUE DO
: 2187. 2257 2 BEGIN
: 2188. 2258 2 LOCAL
: 2189. 2259 2 FLD: REF FLD_BLOCK,
: 2190. 2260 2 KEY: REF KEY_BLOCK,
: 2191. 2261 2 MINKEY: KEY_BLOCK,
: 2192. 2262 2 MINFLD: FLD_BLOCK,
: 2193. 2263 2 COMPAT;
: 2194. 2264 2
: 2195. 2265 2 | Find a current key
: 2196. 2266 2
: 2197. 2267 2 FMT = FMT_ROOT - %FIELDEXPAND(FMT_NEXT,0);
: 2198. 2268 2 KEY = 0;
: 2199. 2269 2 WHILE (FMT = .FMT[FMT_NEXT]) NEQ 0 DO
: 2200. 2270 2 IF (KEY = .FMT[FMT_CKEY]) NEQ 0 THEN EXITLOOP;
: 2201. 2271 2
: 2202. 2272 2 | Exit loop if all keys have been processed
: 2203. 2273 2
: 2204. 2274 2 IF KEY[BASE_] EQL 0 THEN EXITLOOP;
: 2205. 2275 2
: 2206. 2276 2 | Save this key
: 2207. 2277 2
: 2208. 2278 2 MINKEY[KEY_FLD] = MINFLD[BASE_];
: 2209. 2279 2 MINKEY[KEY_ASCE] = .KEY[KEY_ASCE];
: 2210. 2280 2 FLD = .KEY[KEY_FLD]; ! ??? Might this be zero?
: 2211. 2281 2 MINFLD[FLD_SIZE] = .FLD[FLD_SIZE];
: 2212. 2282 2 MINFLD[FLD_DTY] = .FLD[FLD_DTY];
: 2213. 2283 2
: 2214. 2284 2 | Assume that all the keys are compatible
: 2215. 2285 2
: 2216. 2286 2 COMPAT = TRUE;
: 2217. 2287 2
: 2218. 2288 2 | Loop through the rest of the formats, determining compatibility
: 2219. 2289 2 | and minimizing our description of the key.
: 2220. 2290 2
: 2221. 2291 2 WHILE (FMT = .FMT[FMT_NEXT]) NEQ 0 DO
: 2222. 2292 2 IF NOT MINCOMPAT(.FMT[FMT_CKEY], MINKEY[BASE_], MINFLD[FLD_SIZE] )
: 2223. 2293 2 THEN
: 2224. 2294 2 COMPAT = FALSE;
: 2225. 2295 2
: 2226. 2296 2 | Now that we have a short description, loop through all the
```



```
2227 2297 3      ! current keys, 'taking off' this minimal key.
2228 2298 3
2229 2299 3      FMT = FMT_ROOT - %FIELDEXPAND(FMT_NEXT,0);
2230 2300 3      WHILE (FMT = .FMT[FMT_NEXT]) NEQ 0 DO
2231 2301 3      IF (KEY = .FMT[FMT_CKEY]) NEQ 0 THEN      ! More keys for this format?
2232 2302 4          BEGIN
2233 2303 4              LOCAL
2234 2304 4                  T: WORD;      ! Size
2235 2305 4
2236 2306 4              ! Check for compatability of the current key with the minimum key.
2237 2307 4              ! If they are compatible, check that the length hasn't changed.
2238 2308 4
2239 2309 4              FLD = .KEY[KEY_FLD];      ! ??? Might this be zero?
2240 2310 4              IF MINCOMPAT( MINKEY[BASE_], .FMT[FMT_CKEY], T )
2241 2311 4              THEN
2242 2312 5                  BEGIN
2243 2313 5                      IF NOT .COMPAT
2244 2314 5                      THEN
2245 2315 5                          BBLOCK[ NEW_KEY(0,FMT[BASE_]), KEY_LIT ] = 0;
2246 2316 5                          IF .T NEQ .FLD[FLD_SIZE]
2247 2317 5                          THEN
2248 2318 6                              BEGIN
2249 2319 6                                  ! Break the key up into two pieces, by defining the short
2250 2320 6                                  ! key just before the current key, and updating the current
2251 2321 6                                  ! key's length.
2252 2322 6                                  BBLOCK[
2253 2323 6                                      NEW_KEY(
2254 2324 6                                          TREE_INSERT(.FLD[FLD_POS],.T,.FLD[FLD_DTY]),
2255 2325 6                                          FMT[BASE_]), KEY_ASCE] = .KEY[KEY_ASCE];
2256 2326 6                                  FLD[FLD_SIZE] = .FLD[FLD_SIZE] - .T;
2257 2327 6                                  END
2258 2328 6                                  ELSE
2259 2329 6                                      BEGIN
2260 2330 6                                          ! The keys are completely compatible, advance
2261 2331 6                                          FMT[FMT_CKEY] = .KEY[KEY_NEXT];
2262 2332 6                                          END;
2263 2333 6                                  END
2264 2334 6                                  ELSE
2265 2335 6                                      BEGIN
2266 2336 6                                          ! Add a constant key to distinguish the formats
2267 2337 6                                          BBLOCK[ NEW_KEY(0,FMT[BASE_]), KEY_LIT ] = 1;
2268 2338 6                                          END;
2269 2339 6                                      END;
2270 2340 6                                  END;
2271 2341 6                                  ! Initialize all the current key pointers.
2272 2342 6                                  FMT = FMT_ROOT - %FIELDEXPAND(FMT_NEXT,0);
2273 2343 6                                  WHILE (FMT = .FMT[FMT_NEXT]) NEQ 0 DO FMT[FMT_CKEY] = .FMT[FMT_KEYS];
2274 2344 6                                  END;
2275 2345 6                                  END;
2276 2346 6                                  END;
2277 2347 6                                  END;
2278 2348 6                                  END;
2279 2349 6                                  END;
2280 2350 6                                  END;
2281 2351 6                                  END;
2282 2352 6                                  END;
2283 2353 6                                  END;
```



```
2284 2354 2 | Make another pass over the key definitions,  
2285 2355 2 | combining adjacent constant keys.  
2286 2356 2 |  
2287 2357 2 | Although this is not strictly necessary, the output looks terrible  
2288 2358 2 | if the user has many different record formats, each with a key that  
2289 2359 2 | incompatible in a different way.  
2290 2360 2 |  
2291 2361 2 | WHILE TRUE DO  
2292 2362 2 | BEGIN  
2293 2363 2 | LOCAL  
2294 2364 2 |     CKEY:      REF KEY_BLOCK,  
2295 2365 2 |     LIT:      BYTE;  
2296 2366 2 |  
2297 2367 2 | LIT = 0;  
2298 2368 2 |  
2299 2369 2 | ! Advance all the format descriptions to constant keys (if any).  
2300 2370 2 |  
2301 2371 2 | CKEY = 0;  
2302 2372 2 | FMT = FMT_ROOT - %FIELDEXPAND(FMT_NEXT,0);  
2303 2373 2 | WHILE (FMT = .FMT[FMT_NEXT]) NEQ 0 DO  
2304 2374 2 | BEGIN  
2305 2375 2 |     LOCAL KEY: REF KEY_BLOCK;  
2306 2376 2 |     KEY = FMT[FMT_CKEY] - %FIELDEXPAND(KEY_NEXT,0);  
2307 2377 2 |     WHILE (KEY = .KEY[KEY_NEXT]) NEQ 0 DO  
2308 2378 2 |         IF .KEY[KEY_FLD] EQL 0 THEN EXITLOOP;  
2309 2379 2 |         FMT[FMT_CKEY] = .KEY[BASE_];  
2310 2380 2 |         IF KEY[BASE_] NEQ 0 THEN CKEY = KEY[BASE_];  
2311 2381 2 |     END;  
2312 2382 2 | IF CKEY[BASE_] EQL 0 THEN EXITLOOP;      ! All done  
2313 2383 2 |  
2314 2384 2 | WHILE TRUE DO  
2315 2385 2 | BEGIN  
2316 2386 2 | LOCAL  
2317 2387 2 |     CCNT,  
2318 2388 2 |     KEY:      REF KEY_BLOCK;  
2319 2389 2 |  
2320 2390 2 | ! Find the last format with constant keys,  
2321 2391 2 | ! checking that we don't accept non-constant keys.  
2322 2392 2 |  
2323 2393 2 | CKEY = 0;  
2324 2394 2 | FMT = FMT_ROOT - %FIELDEXPAND(FMT_NEXT,0);  
2325 2395 2 | WHILE (FMT = .FMT[FMT_NEXT]) NEQ 0 DO  
2326 2396 2 |     IF (KEY = .FMT[FMT_CKEY]) NEQ 0 THEN  
2327 2397 2 |         IF .KEY[KEY_FLD] EQL 0 THEN CKEY = KEY[BASE_];  
2328 2398 2 |     IF CKEY[BASE_] EQL 0 THEN EXITLOOP;  
2329 2399 2 |  
2330 2400 2 | ! Count the number of constant keys for this format.  
2331 2401 2 |  
2332 2402 2 | CCNT = 1;  
2333 2403 2 | KEY = CKEY[BASE_];  
2334 2404 2 | WHILE (KEY = .KEY[KEY_NEXT]) NEQ 0 DO  
2335 2405 2 |     IF .KEY[KEY_FLD] NEQ 0 THEN EXITLOOP ELSE CCNT = .CCNT + 1;  
2336 2406 2 |  
2337 2407 2 | ! Pass through all the formats, determining which formats  
2338 2408 2 | ! have constant keys equal to this group.  
2339 2409 2 |  
2340 2410 2 | FMT = FMT_ROOT - %FIELDEXPAND(FMT_NEXT,0);
```



```
2341 2411 4 WHILE (FMT = .FMT[FMT_NEXT]) NEQ 0 DO
2342 2412 4 IF (KEY = .FMT[FMT_CKEY]) NEQ 0 THEN
2343 2413 4 IF .KEY[KEY_FLD] EQL 0 THEN
2344 2414 5 BEGIN
2345 2415 5 LOCAL
2346 2416 5 K2: REF KEY_BLOCK;
2347 2417 5 KEY = .FMT[FMT_CKEY];
2348 2418 5 K2 = CKEY[BASE_];
2349 2419 6 IF BEGIN
2350 2420 7 IF NOT (DECR I FROM .CCNT-1 TO 0 DO
2351 2421 8 BEGIN
2352 2422 8 IF KEY[BASE_] EQL 0 THEN EXITLOOP FALSE;
2353 2423 8 IF .KEY[KEY_FLD] NEQ 0 THEN EXITLOOP FALSE;
2354 2424 8 IF .KEY[KEY_LIT] NEQ .K2[KEY_LIT] THEN EXITLOOP FALSE;
2355 2425 8 KEY = .KEY[KEY_NEXT];
2356 2426 8 K2 = .K2[KEY_NEXT];
2357 2427 6 END) THEN FALSE
2358 2428 6 ELIF KEY[BASE_] EQL 0 THEN TRUE
2359 2429 6 ELIF .KEY[KEY_FLD] NEQ 0 THEN TRUE
2360 2430 6 ELSE FALSE
2361 2431 6 END
2362 2432 5 THEN
2363 2433 6 BEGIN
2364 2434 6 |
2365 2435 6 | The constant keys are identical.
2366 2436 6 | Compress these constants, and advance FMT_CKEY past them.
2367 2437 6 | Note that this is why CKEY must be the processed last.
2368 2438 6 |
2369 2439 6 KEY = K2 = .FMT[FMT_CKEY];
2370 2440 6 K2[KEY_LIT] = .LIT;
2371 2441 6 DECR I FROM .CCNT-1 TO 0 DO K2 = .K2[KEY_NEXT];
2372 2442 6 FMT[FMT_CKEY] = KEY[KEY_NEXT] = K2[BASE_];
2373 2443 5 END;
2374 2444 4 END;
2375 2445 4
2376 2446 4
2377 2447 4 ! Go look for some more, this time using a different literal
2378 2448 4 !
2379 2449 4 LIT = .LIT + 1;
2380 2450 4 IF .LIT EQL 0
2381 2451 4 THEN
2382 2452 4 SOR_ERROR(SRTRN$_COMPLEX);
2383 2453 3 END;
2384 2454 3
2385 2455 2 END;
2386 2456 2
2387 2457 2 RETURN;
2388 2458 1 END;
```

```
01FC 0000 DEF_SUBFIELDS:
58 0000' CF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8
57 FEE3 CF 9E 00007 MOVAB FMT_ROOT, R8
MOVAB NEW_KEY, R7
```

: 2240
:
:

			5E		24	C2	0000C		SUBL2	#36, SP		
			53		68	9E	0000F		MOVAB	FMT ROOT, FMT	:	2251
			53		63	D0	00012	1\$:	MOVL	(FMT), FMT	:	2252
					07	13	00015		BEQL	2\$	:	
		08	A3		04	A3	D0	00017	MOVL	4(FMT), 8(FMT)	:	
						F4	11	0001C	BRB	1\$	:	
			53			68	9E	0001E	2\$:	MOVAB	FMT_ROOT, FMT	2267
						54	D4	00021	CLRL	KEY	:	2268
			53			63	D0	00023	3\$:	MOVL	(FMT), FMT	2269
						06	13	00026	BEQL	4\$	:	
			54		08	A3	D0	00028	MOVL	8(FMT), KEY	:	2270
						F5	13	0002C	BEQL	3\$	:	
						54	D5	0002E	4\$:	TSTL	KEY	2274
						03	12	00030	BNEQ	5\$	:	
						00A3	31	00032	BRW	12\$	:	
		1C	AE		04	AE	9E	00035	5\$:	MOVAB	MINFLD, MINKEY+4	2278
20	AE		00		08	A4	F0	0003A	INSV	8(KEY), #0, #1, MINKEY+8	:	2279
			52		04	A4	D0	00041	MOVL	4(KEY), FLD	:	2280
		OE	AE		0A	A2	B0	00045	MOVW	10(FLD), MINFLD+10	:	2281
		14	AE		10	A2	90	0004A	MOVB	16(FLD), MINFLD+16	:	2282
			55			01	D0	0004F	MOVL	#1, COMPAT	:	2286
			53			63	D0	00052	6\$:	MOVL	(FMT), FMT	2291
						15	13	00055	BEQL	7\$	:	
					OE	AE	9F	00057	PUSHAB	MINFLD+10	:	2292
					1C	AE	9F	0005A	PUSHAB	MINKEY	:	
					08	A3	DD	0005D	PUSHL	8(FMT)	:	
		00AF	C7			03	FB	00060	CALLS	#3, MINCOMPAT	:	
			EA			50	E8	00065	BLBS	R0, 6\$	:	
						55	D4	00068	CLRL	COMPAT	:	2294
						E6	11	0006A	BRB	6\$	:	2292
			53			68	9E	0006C	7\$:	MOVAB	FMT ROOT, FMT	2299
			53			63	D0	0006F	8\$:	MOVL	(FMT), FMT	2300
						AA	13	00072	BEQL	2\$	:	
			54		08	A3	D0	00074	MOVL	8(FMT), KEY	:	2301
						F5	13	00078	BEQL	8\$	:	
			52		04	A4	D0	0007A	MOVL	4(KEY), FLD	:	2309
						5E	DD	0007E	PUSHL	SP	:	2310
					08	A3	DD	00080	PUSHL	8(FMT)	:	
					20	AE	9F	00083	PUSHAB	MINKEY	:	
		00AF	C7			03	FB	00086	CALLS	#3, MINCOMPAT	:	
			3D			50	E9	0008B	BLBC	R0, 11\$	:	
			0A			55	E8	0008E	BLBS	COMPAT, 9\$	:	2313
						53	DD	00091	PUSHL	FMT	:	2315
						7E	D4	00093	CLRL	-(SP)	:	
			67			02	FB	00095	CALLS	#2, NEW_KEY	:	
					08	A0	94	00098	CLRB	8(R0)	:	
		0A	A2			6E	B1	0009B	9\$:	CMPW	T, 10(FLD)	2316
						24	13	0009F	BEQL	10\$	:	
						53	DD	000A1	PUSHL	FMT	:	2327
			7E		10	A2	9A	000A3	MOVZBL	16(FLD), -(SP)	:	2326
			7E		08	AE	3C	000A7	MOVZWL	T, -(SP)	:	
			7E		08	A2	3C	000AB	MOVZWL	8(FLD), -(SP)	:	
		41	A7			03	FB	000AF	CALLS	#3, TREE_INSERT	:	
						50	DD	000B3	PUSHL	R0	:	
			67			02	FB	000B5	CALLS	#2, NEW_KEY	:	2327
08	A0		00		08	A4	F0	000B8	INSV	8(KEY), #0, #1, 8(R0)	:	
			A2			6E	A2	000BF	SUBW2	T, 10(FLD)	:	2328

08	A3		AA 11 000C3	BRB 8\$	2316
			64 D0 000C5 10\$:	MOVL (KEY), 8(FMT)	2335
			A4 11 000C9	BRB 8\$	2310
			53 DD 000CB 11\$:	PUSHL FMT	2343
			7E D4 000CD	CLRL -(SP)	
08	67		02 FB 000CF	CALLS #2, NEW KEY	
	A0		01 90 000D2	MOVB #1, 8(R0)	
			97 11 000D6	BRB 8\$	2301
	53		68 9E 000D8 12\$:	MOVAB FMT ROOT, FMT	2350
	53		63 D0 000DB 13\$:	MOVL (FMT), FMT	2351
			07 13 000DE	BEQL 14\$	
08	A3	04	A3 D0 000E0	MOVL 4(FMT), 8(FMT)	
			F4 11 000E5	BRB 13\$	
			56 94 000E7 14\$:	CLRB LIT	2367
			55 D4 000E9	CLRL CKEY	2371
	53		68 9E 000EB	MOVAB FMT ROOT, FMT	2372
	53		63 D0 000EE 15\$:	MOVL (FMT), FMT	2373
			19 13 000F1	BEQL 18\$	
	50	08	A3 9E 000F3	MOVAB 8(FMT), KEY	2376
	50		60 D0 000F7 16\$:	MOVL (KEY), KEY	2377
			05 13 000FA	BEQL 17\$	
		04	A0 D5 000FC	TSTL 4(KEY)	2378
			F6 12 000FF	BNEQ 16\$	
08	A3		50 D0 00101 17\$:	MOVL KEY, 8(FMT)	2379
			E7 13 00105	BEQL 15\$	2380
	55		50 D0 00107	MOVL KEY, CKEY	
			E2 11 0010A	BRB 15\$	2373
			55 D5 0010C 18\$:	TSTL CKEY	2382
			01 12 0010E	BNEQ 19\$	
				RET	
			55 D4 00111 19\$:	CLRL CKEY	2393
	53		68 9E 00113	MOVAB FMT ROOT, FMT	2394
	53		63 D0 00116 20\$:	MOVL (FMT), FMT	2395
			10 13 00119	BEQL 21\$	
	52	08	A3 D0 0011B	MOVL 8(FMT), KEY	2396
			F5 13 0011F	BEQL 20\$	
		04	A2 D5 00121	TSTL 4(KEY)	2397
			F0 12 00124	BNEQ 20\$	
	55		52 D0 00126	MOVL KEY, CKEY	
			EB 11 00129	BRB 20\$	2396
			55 D5 0012B 21\$:	TSTL CKEY	2398
			B8 13 0012D	BEQL 14\$	
	51		01 D0 0012F	MOVL #1, CCNT	2402
	52		55 D0 00132	MOVL CKEY, KEY	2403
	52		62 D0 00135 22\$:	MOVL (KEY), KEY	2404
			09 13 00138	BEQL 23\$	
		04	A2 D5 0013A	TSTL 4(KEY)	2405
			04 12 0013D	BNEQ 23\$	
			51 D6 0013F	INCL CCNT	
			F2 11 00141	BRB 22\$	
	53		68 9E 00143 23\$:	MOVAB FMT ROOT, FMT	2410
	53		63 D0 00146 24\$:	MOVL (FMT), FMT	2411
			58 13 00149	BEQL 30\$	
	52	08	A3 D0 0014B	MOVL 8(FMT), KEY	2412
			F5 13 0014F	BEQL 24\$	
		04	A2 D5 00151	TSTL 4(KEY)	2413
			F0 12 00154	BNEQ 24\$	

SRTTRN
V04-000

K 2
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 88
(28)

	52	08	A3	D0	00156	MOVL	8(FMT), KEY	:	2417
	54		55	D0	0015A	MOVL	CKEY, K2	:	2418
	50		51	D0	0015D	MOVL	CCNT, I	:	2420
			16	11	00160	BRB	26\$	:	
			52	D5	00162	25\$: TSTL	KEY	:	2422
			E0	13	00164	BEQL	24\$	:	
		04	A2	D5	00166	TSTL	4(KEY)	:	2423
			DB	12	00169	BNEQ	24\$	:	
08	A4	08	A2	91	0016B	CMPB	8(KEY), 8(K2)	:	2424
			D4	12	00170	BNEQ	24\$	:	
	52		62	D0	00172	MOVL	(KEY), KEY	:	2425
	54		64	D0	00175	MOVL	(K2), K2	:	2426
	E7		50	F4	00178	26\$: SOBGEQ	I, 25\$	:	2420
			52	D5	0017B	TSTL	KEY	:	2428
			05	13	0017D	BEQL	27\$	:	
		04	A2	D5	0017F	TSTL	4(KEY)	:	2429
			C2	13	00182	BEQL	24\$	:	
	54	08	A3	D0	00184	27\$: MOVL	8(FMT), K2	:	2439
	52		54	D0	00188	MOVL	K2, KEY	:	
08	A4		56	90	0018B	MOVB	LIT, 8(K2)	:	2440
	50		51	D0	0018F	MOVL	CCNT, I	:	2441
			03	11	00192	BRB	29\$	:	
	54		64	D0	00194	28\$: MOVL	(K2), K2	:	
	FA		50	F4	00197	29\$: SOBGEQ	I, 28\$	:	
	62		54	D0	0019A	MOVL	K2, (KEY)	:	2442
08	A3		54	D0	0019D	MOVL	K2, 8(FMT)	:	
			A3	11	001A1	BRB	24\$	:	2413
			56	96	001A3	30\$: INCB	LIT	:	2449
			0B	12	001A5	BNEQ	31\$	:	2450
		001C8092	8F	DD	001A7	PUSHL	#1867922	:	2452
F2F6	C7		01	FB	001AD	CALLS	#1, SOR_ERROR	:	
			FF5C	31	001B2	31\$: BRW	19\$	:	2384
			04	001B5	RET			:	2458

; Routine Size: 438 bytes, Routine Base: \$CODE\$ + 0E1C


```
2390 2459 1 ROUTINE PUT_NAME(PTR: REF FLD_BLOCK): NOVALUE =
2391 2460 2 BEGIN
2392 2461 2
2393 2462 2 ! Since we will issue this name in pieces,
2394 2463 2 ! first make sure it will all fit on one line.
2395 2464 2
2396 2465 2 IF .OUT_DSC[0] LEQ 4+5+1+5 THEN PUT_LINE(1);
2397 2466 2
2398 2467 2 !<s1>
2399 2468 2 SELECTONE .PTR[FLD_DTY] OF
2400 2469 2 SET
2401 2470 2 [DSC$K_DTYPE_B]: PUT_TEXT(LENADR('B_'));
2402 2471 2 [DSC$K_DTYPE_T]: PUT_TEXT(LENADR('T_'));
2403 2472 2 [DSC$K_DTYPE_NRO]: PUT_TEXT(LENADR('NRO_'));
2404 2473 2 [DSC$K_DTYPE_F]: PUT_TEXT(LENADR('F_'));
2405 2474 2 [DSC$K_DTYPE_D]: PUT_TEXT(LENADR('D_'));
2406 2475 2 [DSC$K_DTYPE_NL]: PUT_TEXT(LENADR('NL_'));
2407 2476 2 [DSC$K_DTYPE_NR]: PUT_TEXT(LENADR('NR_'));
2408 2477 2 [DSC$K_DTYPE_NLO]: PUT_TEXT(LENADR('NLO_'));
2409 2478 2 [DSC$K_DTYPE_NU]: PUT_TEXT(LENADR('NU_'));
2410 2479 2 [DSC$K_DTYPE_P]: PUT_TEXT(LENADR('P_'));
2411 2480 2 [DSC$K_DTYPE_NZ]: PUT_TEXT(LENADR('NZ_'));
2412 2481 2 [DSC$K_DTYPE_BU]: PUT_TEXT(LENADR('BU_'));
2413 2482 2 [DSC$K_DTYPE_DIBOL]: PUT_TEXT(LENADR('DIBOL_'));
2414 2483 2 [DSC$K_DTYPE_AZ]: PUT_TEXT(LENADR('AZ_'));
2415 2484 2 [OTHERWISE]: RETURN SOR_ERROR(SRTTRN$_INV_DATA);
2416 2485 2 TES;
2417 2486 2 CVT_BTA(.PTR[FLD_POS]);
2418 2487 2 PUT_TEXT(LENADR(' '));
2419 2488 2 CVT_BTA(.PTR[FLD_SIZ]);
2420 2489 1 END;
```

.PSECT \$PLITS,NOWRT,NOEXE,2

		5F	42	00188	P.ACB:	.ASCII	\B_\	
		5F	54	0018A	P.ACC:	.ASCII	\T_\	
5F	4F	52	4E	0018C	P.ACD:	.ASCII	\NRO_\	
		5F	46	00190	P.ACE:	.ASCII	\F_\	
		5F	44	00192	P.ACF:	.ASCII	\D_\	
		5F	4C	4E	00194	P.ACG:	\NC_\	
		5F	52	4E	00197	P.ACH:	\NR_\	
5F	4F	4C	4E	0019A	P.ACI:	.ASCII	\NLO_\	
		5F	55	4E	0019E	P.ACJ:	\NU_\	
		5F	50	001A1	P.ACK:	.ASCII	\P_\	
		5F	5A	4E	001A3	P.ACL:	\NZ_\	
		5F	55	42	001A6	P.ACM:	\BU_\	
5F	4C	4F	42	49	44	001A9	P.ACN:	\DIBOL_\
		5F	5A	41	001AF	P.ACO:	\AZ_\	
		5F			001B2	P.ACP:	_\	

.PSECT \$CODE\$,NOWRT,2

001C 00000 PUT_NAME:

	54	0000'	CF	9E	00002		.WORD	Save R2,R3,R4		2459
	0F	0000'	CF	D1	00007		MOVAB	P.ACB, R4		
			07	14	0000C		CMPL	OUT_DSC, #15		2465
			01	DD	0000E		BGTR	1\$		
0000V	CF		01	FB	00010		PUSHL	#1		
	53	04	AC	D0	00015	1\$:	CALLS	#1, PUT_LINE		2468
	52	10	A3	9A	00019		MOVL	PTR, R3		
	06		52	91	0001D		MOVZBL	16(R3), R2		2470
			04	12	00020		CMPB	R2, #6		
			54	DD	00022		BNEQ	2\$		
			28	11	00024		PUSHL	R4		
	0E		52	91	00026	2\$:	BRB	7\$		2471
			05	12	00029		CMPB	R2, #14		
		02	A4	9F	0002B		BNEQ	3\$		
			1E	11	0002E		PUSHAB	P.ACC		
	13		52	91	00030	3\$:	BRB	7\$		2472
			07	12	00033		CMPB	R2, #19		
		04	A4	9F	00035		BNEQ	5\$		
			04	DD	00038	4\$:	PUSHAB	P.ACD		
			72	11	0003A		PUSHL	#4		
	0A		52	91	0003C	5\$:	BRB	18\$		2473
			05	12	0003F		CMPB	R2, #10		
		08	A4	9F	00041		BNEQ	6\$		
			08	11	00044		PUSHAB	P.ACE		
	0B		52	91	00046	6\$:	BRB	7\$		2474
			07	12	00049		CMPB	R2, #11		
		0A	A4	9F	0004B		BNEQ	8\$		
			02	DD	0004E	7\$:	PUSHAB	P.ACF		
			5C	11	00050		PUSHL	#2		
	10		52	91	00052	8\$:	BRB	18\$		2475
			05	12	00055		CMPB	R2, #16		
		0C	A4	9F	00057		BNEQ	9\$		
			50	11	0005A		PUSHAB	P.ACG		
	12		52	91	0005C	9\$:	BRB	17\$		2476
			05	12	0005F		CMPB	R2, #18		
		0F	A4	9F	00061		BNEQ	10\$		
			46	11	00064		PUSHAB	P.ACH		
	11		52	91	00066	10\$:	BRB	17\$		2477
			05	12	00069		CMPB	R2, #17		
		12	A4	9F	0006B		BNEQ	11\$		
			C8	11	0006E		PUSHAB	P.ACI		
	0F		52	91	00070	11\$:	BRB	4\$		2478
			05	12	00073		CMPB	R2, #15		
		16	A4	9F	00075		BNEQ	12\$		
			32	11	00078		PUSHAB	P.ACJ		
	15		52	91	0007A	12\$:	BRB	17\$		2479
			05	12	0007D		CMPB	R2, #21		
		19	A4	9F	0007F		BNEQ	13\$		
			CA	11	00082		PUSHAB	P.ACK		
	14		52	91	00084	13\$:	BRB	7\$		2480
			05	12	00087		CMPB	R2, #20		
		1B	A4	9F	00089		BNEQ	14\$		
			1E	11	0008C		PUSHAB	P.ACL		
	02		52	91	0008E	14\$:	BRB	17\$		2481
			05	12	00091		CMPB	R2, #2		
		1E	A4	9F	00093		BNEQ	15\$		
							PUSHAB	P.ACM		

SRTTRN
V04-000

N 2
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 91
(29)

			14	11	00096		BRB	17\$		
	28		52	91	00098	15\$:	CMPB	R2, #40		2482
			07	12	0009B		BNEQ	16\$		
		21	A4	9F	0009D		PUSHAB	P.ACN		
			06	DD	000A0		PUSHL	#6		
			0A	11	000A2		BRB	18\$		
	29		52	91	000A4	16\$:	CMPB	R2, #41		2483
			0C	12	000A7		BNEQ	19\$		
		27	A4	9F	000A9		PUSHAB	P.ACO		
			03	DD	000AC	17\$:	PUSHL	#3		
0000V	CF		02	FB	000AE	18\$:	CALLS	#2, PUT_TEXT		
			0C	11	000B3		BRB	20\$		
		001C801C	8F	DD	000B5	19\$:	PUSHL	#1867804		2484
EF6E	CF		01	FB	000BB		CALLS	#1, SOR_ERROR		
				04	000C0		RET			
	7E	08	A3	3C	000C1	20\$:	MOVZWL	8(R3), -(SP)		2486
0000V	CF		01	FB	000C5		CALLS	#1, CVT_BTA		
		2A	A4	9F	000CA		PUSHAB	P.ACP		2487
			01	DD	000CD		PUSHL	#1		
0000V	CF		02	FB	000CF		CALLS	#2, PUT_TEXT		
	7E	0A	A3	3C	000D4		MOVZWL	10(R3), -(SP)		2488
0000V	CF		01	FB	000D8		CALLS	#1, CVT_BTA		
				04	000DD		RET			2489

; Routine Size: 222 bytes, Routine Base: \$CODE\$ + 0FD2


```
2422 1 ROUTINE PUT_FFLD(PTR: REF FLD_BLOCK, NESTED): NOVALUE =
2423 BEGIN
2424 LOCAL
2425 FFLD: REF FFLD_BLOCK;
2426
2427 FFLD = .PTR[FLD_FFLD];
2428
2429 WHILE .FFLD[FFLD_NEXT] NEQ 0 DO
2430 BEGIN
2431 PUT_TEXT(
2432 LENADR(%CHAR(9),X IF,' COND '),
2433 3, CVT_BTO(.FFLD[FFLD_COND]));
2434 PUT_TEXT(
2435 LENADR(' ',X THEN,' %0'),
2436 3, CVT_BTO(.FFLD[FFLD_LIT]));
2437 LENADR(' ',X ELSE));
2438 PUT_LINE(.NESTED);
2439 FFLD = .FFLD[FFLD_NEXT];
2440 END;
2441
2442 PUT_TEXT(LENADR(%CHAR(9),'%0'),
2443 3, CVT_BTO(.FFLD[FFLD_LIT]));
2444
2445 END;
```

```
.PSECT $SPLITS,NOWRT,NOEXE,2
5F 44 4E 4F 43 20 46 49 09 001B3 P.ACQ: .ASCII <9>\IF COND\_
4F 25 20 4E 45 48 54 20 001BC P.ACR: .ASCII \ THEN %0\
45 53 4C 45 20 001C4 P.ACS: .ASCII \ ELSE\
4F 25 09 001C9 P.ACT: .ASCII <9>\%0\
```

```
.PSECT $CODE$,NOWRT,2
001C 00000 PUT_FFLD:
54 F546 CF 9E 00002 .WORD Save R2,R3,R4 : 2490
53 0000V CF 9E 00007 MOVAB CVT_BTO, R4 :
50 04 AC D0 0000C MOVAB PUT_TEXT, R3 :
52 0C A0 D0 0001C MOVL PTR, R0 : 2495
62 D5 00014 1$: MOVL 12(R0), FFLD :
3B 13 00016 TSTL (FFLD) : 2497
7E 04 A2 3C 00018 BEQL 2$ :
64 01 FB 0001C MOVZWL 4(FFLD), -(SP) : 2501
50 DD 0001F CALLS #1, CVT_BTO
03 DD 00021 PUSHL R0
0000' CF 9F 00023 PUSHL #3 : 2499
09 DD 00027 PUSHAB P.ACQ : 2500
63 04 FB 00029 PUSHL #9 : 2499
0000' CF 9F 0002C CALLS #4, PUT_TEXT :
05 DD 00030 PUSHAB P.ACS : 2505
7E 06 A2 9A 00032 PUSHL #5 : 2502
64 01 FB 00036 MOVZBL 6(FFLD), -(SP) : 2504
CALLS #1, CVT_BTO
```



```

C 3
16-Sep-1984 01:11:52 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:54 [SORT32.SRC]SRTTRN.B32;1

```

Page 93
(30)

		50	DD	00039	PUSHL	R0	:	
		03	DD	0003B	PUSHL	#3	:	2502
	0000'	CF	9F	0003D	PUSHAB	P.ACR	:	2503
		08	DD	00041	PUSHL	#8	:	2502
63		06	FB	00043	CALLS	#6, PUT_TEXT	:	
	08	AC	DD	00046	PUSHL	NESTED	:	2506
0000V	CF	01	FB	00049	CALLS	#1, PUT_LINE	:	
	52	62	D0	0004E	MOVL	(FFLD), -FFLD	:	2507
		C1	11	00051	BRB	1\$	:	2497
7E		06	A2	9A 00053	MOVZBL	6(FFLD), -(SP)	:	2511
64		01	FB	00057	CALLS	#1, CVT_BT0	:	
		50	DD	0005A	PUSHL	R0	:	
		03	DD	0005C	PUSHL	#3	:	2510
	0000'	CF	9F	0005E	PUSHAB	P.ACT	:	
		03	DD	00062	PUSHL	#3	:	
63		04	FB	00064	CALLS	#4, PUT_TEXT	:	
		04	00067	RET			:	2513

```
; Routine Size: 104 bytes,    Routine Base: $CODE$ + 10B0
```

SRT
V04


```
: 2447      2514 1 ROUTINE PUT_DEFN(PTR: REF FLD_BLOCK): NOVALUE =
: 2448      2515 2 BEGIN
: 2449      2516 2 PUT_TEXT(LENADR('/',X_FIEL,'=(' ,X_NAME,':')));
: 2450      2517 2 PUT_NAME(PTR[BASE]);
: 2451      2518 2 PUT_TEXT(LENADR(' ',X_POSI,':'));
: 2452      2519 2 CVT_BTA(.PTR[FLD_POS]);
: 2453      2520 2 !<st>
: 2454      2521 2 SELECTONE .PTR[FLD_DTY] OF
: 2455      2522 2 SET
: 2456      2523 2 [DSC$K_DTYPE_B,
: 2457      2524 2 DSC$K_DTYPE_T,
: 2458      2525 2 DSC$K_DTYPE_BU,
: 2459      2526 2 DSC$K_DTYPE_F,
: 2460      2527 2 DSC$K_DTYPE_AZ,
: 2461      2528 2 DSC$K_DTYPE_D]; PUT_TEXT(LENADR(' ',X_SIZE,':'));
: 2462      2529 2 [DSC$K_DTYPE_NRO,
: 2463      2530 2 DSC$K_DTYPE_NL,
: 2464      2531 2 DSC$K_DTYPE_NR,
: 2465      2532 2 DSC$K_DTYPE_NLO,
: 2466      2533 2 DSC$K_DTYPE_NU,
: 2467      2534 2 DSC$K_DTYPE_P,
: 2468      2535 2 DSC$K_DTYPE_DIBOL,
: 2469      2536 2 DSC$K_DTYPE_NZ]; PUT_TEXT(LENADR(' ',X_DIGI,':'));
: 2470      2537 2 [OTHERWISE]: RETURN SOR_ERROR(SRTTRN$_INV_DATA);
: 2471      2538 2 TES;
: 2472      2539 2 CVT_BTA(.PTR[FLD_SIZ]);
: 2473      2540 2 PUT_TEXT(LENADR(' ',X_SIZ,':'));
: 2474      2541 2 !<st>
: 2475      2542 2 SELECTONE .PTR[FLD_DTY] OF
: 2476      2543 2 SET
: 2477      2544 2 [DSC$K_DTYPE_B]: PUT_TEXT(LENADR(X_BINA,':'));
: 2478      2545 2 [DSC$K_DTYPE_T]: PUT_TEXT(LENADR(X_CHAR,':'));
: 2479      2546 2 [DSC$K_DTYPE_NRO]: PUT_TEXT(LENADR(X_DEC1,':',X_TRAI,':',X_OVER,':'));
: 2480      2547 2 [DSC$K_DTYPE_F]: PUT_TEXT(LENADR(X_FFLT,':'));
: 2481      2548 2 [DSC$K_DTYPE_D]: PUT_TEXT(LENADR(X_DFLT,':'));
: 2482      2549 2 [DSC$K_DTYPE_NL]: PUT_TEXT(LENADR(X_DEC1,':',X_LEAD,':',X_SEPA,':'));
: 2483      2550 2 [DSC$K_DTYPE_NR]: PUT_TEXT(LENADR(X_DEC1,':',X_TRAI,':',X_SEPA,':'));
: 2484      2551 2 [DSC$K_DTYPE_NLO]: PUT_TEXT(LENADR(X_DEC1,':',X_LEAD,':',X_OVER,':'));
: 2485      2552 2 [DSC$K_DTYPE_NU]: PUT_TEXT(LENADR(X_DEC1,':',X_UNSI,':'));
: 2486      2553 2 [DSC$K_DTYPE_P]: PUT_TEXT(LENADR(X_PACK,':'));
: 2487      2554 2 [DSC$K_DTYPE_AZ]: PUT_TEXT(LENADR(X_ASCZ,':'));
: 2488      2555 2 [DSC$K_DTYPE_NZ]: PUT_TEXT(LENADR(X_ZONE,':'));
: 2489      2556 2 [DSC$K_DTYPE_BU]: PUT_TEXT(LENADR(X_BINA,':',X_UNSI,':'));
: 2490      2557 2 [DSC$K_DTYPE_DIBOL]: PUT_TEXT(LENADR(X_DIBO,':'));
: 2491      2558 2 [OTHERWISE]: RETURN SOR_ERROR(SRTTRN$_INV_DATA);
: 2492      2559 2 TES;
: 2493      2560 2
: 2494      2561 2 RETURN TRUE;
: 2495      2562 2
: 2496      2563 1 END;
```

.PSECT \$SPLITS,NOWRT,NOEXE,2

```
3A 45 4D 41 4E 28 3D 44 4C 45 49 46 2F 001CC P.ACU: .ASCII \ /FIELD=(NAME:\
3A 4E 4F 49 54 49 53 4F 50 20 2C 001D9 P.ACV: .ASCII \, POSITION:\
```


4E	49	4C	49	41	29	52	45	29	59	52	41	4E	49	53	20	2C	001E4	P.ACW:	.ASCII	\, SIZE:\	
43	4E	55	50	52	45	54	2C	4C	41	4D	49	43	45	44	20	2C	001EB	P.ACX:	.ASCII	\, DIGITS:\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	001F4	P.ACY:	.ASCII	\, \	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	001F6	P.ACZ:	.ASCII	\(BINARY)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	001FD	P.ADA:	.ASCII	\(CHARACTER)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	00207	P.ADB:	.ASCII	\(DECIMAL,TRAILING_SIGN,OVERPUNCHED_SIGN)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	00216				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	00225				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	0022E	P.ADC:	.ASCII	\(F_FLOATING)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	00239	P.ADD:	.ASCII	\(D_FLOATING)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	00244	P.ADE:	.ASCII	\(DECIMAL,LEADING_SIGN,SEPARATE_SIGN)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	00253				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	00262				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	00267	P.ADF:	.ASCII	\(DECIMAL,TRAILING_SIGN,SEPARATE_SIGN)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	00276				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	00285				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	0028B	P.ADG:	.ASCII	\(DECIMAL,LEADING_SIGN,OVERPUNCHED_SIGN)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	0029A				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	002A9				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	002B1	P.ADH:	.ASCII	\(DECIMAL,UNSIGNED)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	002C0				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	002C2	P.ADI:	.ASCII	\(PACKED_DECIMAL)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	002D1	P.ADJ:	.ASCII	\(ASCII_ZONED)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	002D9	P.ADK:	.ASCII	\(ZONED)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	002DD	P.ADL:	.ASCII	\(BINARY,UNSIGNED)\	
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	002E3				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	002F2				
				29	47	4E	49	54	41	4F	4C	46	5F	44	45	48	002F3	P.ADM:	.ASCII	\(DIBOL)\	

.PSECT \$CODE\$,NOWRT,2

003C 00000 PUT_DEFN:

	55	0000V	CF	9E	00002	.WORD	Save R2,R3,R4,R5	2514
	54	0000V	CF	9E	00007	MOVAB	PUT TEXT, R5	
			54	DD	0000C	MOVAB	P.ACX, R4	
			0D	DD	0000E	PUSHL	R4	2516
	65		02	FB	00010	PUSHL	#13	
	52	04	AC	DD	00013	CALLS	#2, PUT_TEXT	
			52	DD	00017	MOVL	PTR, R2	2517
FE9C	CF		01	FB	00019	PUSHL	R2	
		0D	A4	9F	0001E	CALLS	#1, PUT_NAME	
			0B	DD	00021	PUSHAB	P.ACX	2518
	65		02	FB	00023	PUSHL	#11	
	7E	08	A2	3C	00026	CALLS	#2, PUT_TEXT	
0000V	CF		01	FB	0002A	MOVZWL	8(R2), =(SP)	2519
	53	10	A2	9A	0002F	CALLS	#1, CVT_BTA	
	02		53	91	00033	MOVZBL	16(R2), -R3	2521
			19	13	00036	CMPB	R3, #2	2523
	06		53	91	00038	BEQL	2\$,	
			14	13	0003B	CMPB	R3, #6	
	0A		53	91	0003D	BEQL	2\$,	
			05	1F	00040	CMPB	R3, #10	
	0B		53	91	00042	BLSSU	1\$,	
			0A	1B	00045	CMPB	R3, #11	
	0E		53	91	00047	BLEQU	2\$,	
					1\$:	CMPB	R3, #14	

		05	13	0004A	BEQL	2\$		
29		53	91	0004C	CMPB	R3, #41		
		07	12	0004F	BNEQ	3\$		
	18	A4	9F	00051	2\$: PUSHAB	P.ACW		2528
		07	DD	00054	PUSHL	#7		
		17	11	00056	BRB	6\$		
0F		53	91	00058	3\$: CMPB	R3, #15		2529
		05	1F	0005B	BLSSU	4\$		
15		53	91	0005D	CMPB	R3, #21		
		08	1B	00060	BLEQU	5\$		
28		53	91	00062	4\$: CMPB	R3, #40		
		03	13	00065	BEQL	5\$		
		00C7	31	00067	BRW	25\$		
	1F	A4	9F	0006A	5\$: PUSHAB	P.ACX		2536
		09	DD	0006D	PUSHL	#9		
65		02	FB	0006F	6\$: CALLS	#2, PUT_TEXT		
7E	0A	A2	3C	00072	MOVZWL	10(R2), -(SP)		2539
CF		01	FB	00076	CALLS	#1, CVT_BTA		
	28	A4	9F	0007B	PUSHAB	P.ACY		2540
		02	DD	0007E	PUSHL	#2		
65		02	FB	00080	CALLS	#2, PUT_TEXT		
06		53	91	00083	CMPB	R3, #6		2544
		07	12	00086	BNEQ	7\$		
	2A	A4	9F	00088	PUSHAB	P.ACZ		
		07	DD	0008B	PUSHL	#7		
		79	11	0008D	BRB	18\$		
0E		53	91	0008F	7\$: CMPB	R3, #14		2545
		07	12	00092	BNEQ	8\$		
	31	A4	9F	00094	PUSHAB	P.ADA		
		0A	DD	00097	PUSHL	#10		
		6D	11	00099	BRB	18\$		
13		53	91	0009B	8\$: CMPB	R3, #19		2546
		07	12	0009E	BNEQ	9\$		
	3B	A4	9F	000A0	PUSHAB	P.ADB		
		27	DD	000A3	PUSHL	#39		
		79	11	000A5	BRB	21\$		
0A		53	91	000A7	9\$: CMPB	R3, #10		2547
		05	12	000AA	BNEQ	10\$		
	62	A4	9F	000AC	PUSHAB	P.ADC		
		08	11	000AF	BRB	11\$		
0B		53	91	000B1	10\$: CMPB	R3, #11		2548
		07	12	000B4	BNEQ	12\$		
	6D	A4	9F	000B6	PUSHAB	P.ADD		
		0B	DD	000B9	PUSHL	#11		
		70	11	000BB	BRB	24\$		
10		53	91	000BD	12\$: CMPB	R3, #16		2549
		07	12	000C0	BNEQ	13\$		
	78	A4	9F	000C2	PUSHAB	P.ADE		
		23	DD	000C5	PUSHL	#35		
		64	11	000C7	BRB	24\$		
12		53	91	000C9	13\$: CMPB	R3, #18		2550
		08	12	000CC	BNEQ	14\$		
	009B	C4	9F	000CE	PUSHAB	P.ADF		
		24	DD	000D2	PUSHL	#36		
		57	11	000D4	BRB	24\$		
11		53	91	000D6	14\$: CMPB	R3, #17		2551
		08	12	000D9	BNEQ	15\$		

	00BF	C4	9F	000DB		PUSHAB	P,ADG		
		26	DD	000DF		PUSHL	#38		
		4A	11	000E1		BRB	24\$		
0F		53	91	000E3	15\$:	CMPB	R3, #15		2552
		08	12	000E6		BNEQ	16\$		
	00E5	C4	9F	000E8		PUSHAB	P,ADH		
		11	DD	000EC		PUSHL	#17		
		3D	11	000EE		BRB	24\$		
15		53	91	000F0	16\$:	CMPB	R3, #21		2553
		08	12	000F3		BNEQ	17\$		
	00F6	C4	9F	000F5		PUSHAB	P,ADI		
		0F	DD	000F9		PUSHL	#15		
		30	11	000FB		BRB	24\$		
29		53	91	000FD	17\$:	CMPB	R3, #41		2554
		08	12	00100		BNEQ	19\$		
	0105	C4	9F	00102		PUSHAB	P,ADJ		
		0C	DD	00106		PUSHL	#12		
		23	11	00108	18\$:	BRB	24\$		
14		53	91	0010A	19\$:	CMPB	R3, #20		2555
		06	12	0010D		BNEQ	20\$		
	0111	C4	9F	0010F		PUSHAB	P,ADK		
		16	11	00113		BRB	23\$		
02		53	91	00115	20\$:	CMPB	R3, #2		2556
		08	12	00118		BNEQ	22\$		
	0117	C4	9F	0011A		PUSHAB	P,ADL		
		10	DD	0011E		PUSHL	#16		
		0B	11	00120	21\$:	BRB	24\$		
28		53	91	00122	22\$:	CMPB	R3, #40		2557
		0A	12	00125		BNEQ	25\$		
	0127	C4	9F	00127		PUSHAB	P,ADM		
		06	DD	0012B	23\$:	PUSHL	#6		
65		02	FB	0012D	24\$:	CALLS	#2, PUT_TEXT		
			04	00130		RET			
	001C801C	8F	DD	00131	25\$:	PUSHL	#1867804		2558
EDAC	CF	01	FB	00137		CALLS	#1, SOR_ERROR		
			04	0013C		RET			2563

; Routine Size: 317 bytes, Routine Base: \$CODE\$ + 1118

```
: 2498 2564 1 ROUTINE GET_DEFN(IN) =
: 2499 2565 1
: 2500 2566 1 ++
: 2501 2567 1
: 2502 2568 1 FUNCTIONAL DESCRIPTION:
: 2503 2569 1
: 2504 2570 1     Gets the definition for a field.
: 2505 2571 1
: 2506 2572 1 FORMAL PARAMETERS:
: 2507 2573 1
: 2508 2574 1     IN     Index into the input buffer for start location of field.
: 2509 2575 1
: 2510 2576 1 IMPLICIT INPUTS:
: 2511 2577 1
: 2512 2578 1     NONE
: 2513 2579 1
: 2514 2580 1 IMPLICIT OUTPUTS:
: 2515 2581 1
: 2516 2582 1     NONE
: 2517 2583 1
: 2518 2584 1 ROUTINE VALUE:
: 2519 2585 1
: 2520 2586 1     Number identifying the field
: 2521 2587 1
: 2522 2588 1 SIDE EFFECTS:
: 2523 2589 1
: 2524 2590 1     NONE
: 2525 2591 1 --
: 2526 2592 2 BEGIN
: 2527 2593 2 LOCAL
: 2528 2594 2     POS,      ! Position
: 2529 2595 2     SIZ,      ! Size
: 2530 2596 2     DTY;      ! Datatype
: 2531 2597 2
: 2532 2598 2 ! Get the value from the 'to' field
: 2533 2599 2
: 2534 2600 2 IF NOT CVT_DTB(4, INP_BUF[.IN+4], SIZ)
: 2535 2601 2 THEN
: 2536 2602 2     SOR_ERROR(SRTTRN$_INV_LIMIT);
: 2537 2603 2
: 2538 2604 2 ! Look for a value in the 'from' field
: 2539 2605 2
: 2540 2606 2 IF CH$FAIL(CH$FIND_NOT_CH(4, INP_BUF[.IN], %C' '))
: 2541 2607 2 THEN
: 2542 2608 2     BEGIN
: 2543 2609 2         ! We have a one-byte field
: 2544 2610 2
: 2545 2611 2         POS = .SIZ;
: 2546 2612 2     END
: 2547 2613 2 ELSE
: 2548 2614 2     BEGIN
: 2549 2615 2         ! Convert the position to binary
: 2550 2616 2
: 2551 2617 2         IF NOT CVT_DTB(4, INP_BUF[.IN], POS)
: 2552 2618 2     THEN
: 2553 2619 2
: 2554 2620 2
```



```
2555 2621 3
2556 2622
2557 2623
2558 2624
2559 2625
2560 2626
2561 2627
2562 2628
2563 2629
2564 2630
2565 2631
2566 2632
2567 2633
2568 2634
2569 2635
2570 2636
2571 2637
2572 2638
2573 2639
2574 2640
2575 2641
2576 2642
2577 2643
2578 2644
2579 2645
2580 2646
2581 2647
2582 2648
2583 2649
2584 2650
2585 2651
2586 2652
2587 2653
2588 2654
2589 2655
2590 2656
2591 2657
2592 2658
2593 2659
2594 2660
2595 2661
2596 2662
2597 2663
2598 2664
2599 2665
2600 2666
2601 2667
2602 2668
2603 2669
2604 2670
2605 2671
2606 2672
2607 2673
2608 2674
2609 2675

      SOR_ERROR(SRTTRN$_INV_LIMIT);
    END;

    ! Compute the size of this field
    IF .SIZ GEQ .POS
    THEN
      SIZ = .SIZ - .POS + 1
    ELSE
      BEGIN
        SOR_ERROR(SRTTRN$_INV_LIMIT);
        SIZ = .POS - .SIZ + 1;
      END;

    ! Determine the datatype
    ! SELECTONE .INP_BUF[7] OF
    SET
    ['B']: DTY = DSC$K_DTYPE_B;
    ['C']: DTY = DSC$K_DTYPE_T;
    ['D']: DTY = DSC$K_DTYPE_NRO;
    ['F']: DTY = (IF .SIZ EQ 8 THEN DSC$K_DTYPE_D ELSE DSC$K_DTYPE_F);
    ['I']: BEGIN
      DTY = DSC$K_DTYPE_NL;
      SIZ = .SIZ - 1;
    END;
    ['J']: BEGIN
      DTY = DSC$K_DTYPE_NR;
      SIZ = .SIZ - 1;
    END;
    ['K']: DTY = DSC$K_DTYPE_NLO;
    ['L']: DTY = DSC$K_DTYPE_DIBOL;
    ['P']: BEGIN
      DTY = DSC$K_DTYPE_P;
      ! SORT-32 incorrectly determines the size of the packed number
      ! SORT-11 does this correctly
      IF NOT .SWITCH_S32 THEN SIZ = .SIZ * 2 - 1;
    END;
    ['Z']: BEGIN
      ! SORT-32 considered this to be zoned
      ! SORT-11 considered this to be ascii zoned
      IF .SWITCH_S32
      THEN DTY = DSC$K_DTYPE_NZ
      ELSE DTY = DSC$K_DTYPE_AZ;
    END;
    [OTHERWISE]: RETURN SOR_ERROR(SRTTRN$_INV_DATA);
  TES;

  RETURN TREE_INSERT(.POS, .SIZ, .DTY);
END;
```

				000C 00000 GET_DEFN:			
	53	EDA5	CF 9E	00002	.WORD	Save R2,R3	2564
	5E		08 C2	00007	MOVAB	SOR_ERROR, R3	
			5E DD	0000A	SUBL2	#8, -SP	
	50	0000'	CF 9E	0000C	PUSHL	SP	2600
		04 BC40	04 9F	00011	MOVAB	INP_BUF+4, R0	
			04 DD	00015	PUSHAB	@IN[R0]	
0000V	CF		03 FB	00017	PUSHL	#4	
	09		50 E8	0001C	CALLS	#3, CVT_DTB	
		001C802A	8F DD	0001F	BLBS	R0, 1\$	
	63		01 FB	00025	PUSHL	#1867818	2602
	52	0000'	CF 9E	00028	CALLS	#1, SOR_ERROR	
	04		20 3B	0002D	MOVAB	INP_BUF, R2	2606
04 BC42			02 12	00033	SKPC	#32, #4, @IN[R2]	
			51 D4	00035	BNEQ	2\$	
			51 D5	00037	CLRL	R1	
			06 12	00039	TSTL	R1	
	04	AE	6E D0	0003B	BNEQ	3\$	
			1A 11	0003F	MOVL	SIZ, POS	2612
		04	AE 9F	00041	BRB	4\$	2606
		04 BC42	04 9F	00044	PUSHAB	POS	2619
			04 DD	00048	PUSHAB	@IN[R2]	
0000V	CF		03 FB	0004A	PUSHL	#4	
	09		50 E8	0004F	CALLS	#3, CVT_DTB	
		001C802A	8F DD	00052	BLBS	R0, 4\$	
	63		01 FB	00058	PUSHL	#1867818	2621
	04	AE	6E D1	0005B	CALLS	#1, SOR_ERROR	
			07 19	0005F	CMPL	SIZ, POS	2626
50		04	AE C3	00061	BLSS	5\$	
			0E 11	00066	SUBL3	POS, SIZ, R0	2628
		001C802A	8F DD	00068	BRB	6\$	
	63		01 FB	0006E	PUSHL	#1867818	2631
50	04		6E C3	00071	CALLS	#1, SOR_ERROR	
		01	A0 9E	00076	SUBL3	SIZ, POS, R0	2632
	50	0000'	CF 9A	0007A	MOVAB	1(R0), SIZ	
	42		50 91	0007F	MOVZBL	INP_BUF+7, R0	2637
			05 12	00083	CMPL	R0, #66	2639
	52		06 D0	00085	BNEQ	7\$	
			7E 11	00088	MOVL	#6, DTY	
	43	8F	50 91	0008A	BRB	18\$	
			05 12	0008E	CMPB	R0, #67	2640
	52		0E D0	00090	BNEQ	8\$	
			78 11	00093	MOVL	#14, DTY	
	44	8F	50 91	00095	BRB	20\$	
			05 12	00099	CMPB	R0, #68	2641
	52		13 D0	0009B	BNEQ	9\$	
			79 11	0009E	MOVL	#19, DTY	
	46	8F	50 91	000A0	BRB	22\$	
			0F 12	000A4	CMPB	R0, #70	2642
	08		6E D1	000A6	BNEQ	11\$	
			05 12	000A9	CMPL	SIZ, #8	
	52		0B D0	000AB	BNEQ	10\$	
			69 11	000AE	MOVL	#11, DTY	
					BRB	22\$	

	52		0A	D0	000B0	10\$:	MOVL	#10, DTY		
			64	11	000B3		BRB	22\$		
49	8F		50	91	000B5	11\$:	CMPB	R0, #73	2643	
			05	12	000B9		BNEQ	12\$		
	52		10	D0	000BB		MOVL	#16, DTY	2644	
			36	11	000BE		BRB	16\$	2645	
4A	8F		50	91	000C0	12\$:	CMPB	R0, #74	2647	
			05	12	000C4		BNEQ	13\$		
	52		12	D0	000C6		MOVL	#18, DTY	2648	
			28	11	000C9		BRB	16\$	2649	
4B	8F		50	91	000CB	13\$:	CMPB	R0, #75	2651	
			05	12	000CF		BNEQ	14\$		
	52		11	D0	000D1		MOVL	#17, DTY		
			43	11	000D4		BRB	22\$		
4C	8F		50	91	000D6	14\$:	CMPB	R0, #76	2652	
			05	12	000DA		BNEQ	15\$		
	52		28	D0	000DC		MOVL	#40, DTY		
			38	11	000DF		BRB	22\$		
50	8F		50	91	000E1	15\$:	CMPB	R0, #80	2653	
			13	12	000E5		BNEQ	17\$		
	52		15	D0	000E7		MOVL	#21, DTY	2654	
	2A	0000'	CF	E8	000EA		BLBS	SWITCH S32, 22\$	2659	
	50		6E	D0	000EF		MOVL	SIZ, R0		
6E	50		01	78	000F2		ASHL	#1, R0, SIZ		
			6E	D7	000F6	16\$:	DECL	SIZ		
			1F	11	000F8		BRB	22\$	2637	
5A	8F		50	91	000FA	17\$:	CMPB	R0, #90	2661	
			0F	12	000FE		BNEQ	21\$		
	05	0000'	CF	E9	00100		BLBC	SWITCH S32, 19\$	2666	
	52		14	D0	00105		MOVL	#20, DTY	2667	
			0F	11	00108	18\$:	BRB	22\$		
	52		29	D0	0010A	19\$:	MOVL	#41, DTY	2668	
			0A	11	0010D	20\$:	BRB	22\$	2637	
		001C801C	8F	DD	0010F	21\$:	PUSHL	#1867804	2670	
	63		01	FB	00115		CALLS	#1, SOR_ERROR		
				04	00118		RET			
			52	DD	00119	22\$:	PUSHL	DTY	2673	
		04	AE	DD	0011B		PUSHL	SIZ		
		0C	AE	DD	0011E		PUSHL	POS		
0D4B	C3		03	FB	00121		CALLS	#3, TREE_INSERT		
			04	00126			RET		2675	

; Routine Size: 295 bytes, Routine Base: \$CODE\$ + 1255

```
: 2611      2676 1 ROUTINE GET_FFLD(LIT, COND) =
: 2612      2677 1
: 2613      2678 1 ++
: 2614      2679 1
: 2615      2680 1 FUNCTIONAL DESCRIPTION:
: 2616      2681 1
: 2617      2682 1     Create a FFLD_BLOCK
: 2618      2683 1
: 2619      2684 1 FORMAL PARAMETERS:
: 2620      2685 1
: 2621      2686 1     LIT      Value for FFLD_LIT
: 2622      2687 1     COND     Value for FFLD_COND
: 2623      2688 1
: 2624      2689 1 IMPLICIT INPUTS:
: 2625      2690 1
: 2626      2691 1     NONE
: 2627      2692 1
: 2628      2693 1 IMPLICIT OUTPUTS:
: 2629      2694 1
: 2630      2695 1     NONE
: 2631      2696 1
: 2632      2697 1 ROUTINE VALUE:
: 2633      2698 1
: 2634      2699 1     Address of FFLD_BLOCK
: 2635      2700 1
: 2636      2701 1 SIDE EFFECTS:
: 2637      2702 1
: 2638      2703 1     NONE
: 2639      2704 1 --
: 2640      2705 2 BEGIN
: 2641      2706 2 LOCAL
: 2642      2707 2     FFLD:  REF FFLD_BLOCK;
: 2643      2708 2
: 2644      2709 2     GET_VM(FFLD_K_SIZE, FFLD);
: 2645      2710 2     FFLD[FFLD_COND] = .COND;
: 2646      2711 2     FFLD[FFLD_LIT] = .LIT;
: 2647      2712 2     RETURN FFLD[BASE_];
: 2648      2713 2
: 2649      2714 1 END;
```

				0000 00000 GET_FFLD:			
				04 C2 00002	.WORD	Save nothing	: 2676
	5E			5E DD 00005	SUBL2	#4, SP	: 2709
				07 DD 00007	PUSHL	SP	:
				02 FB 00009	PUSHL	#7	:
F937	CF			6E D0 0000E	CALLS	#2, GET_VM	:
	50			04 AC B0 00011	MOVL	FFLD, R0	: 2710
04	A0	08		04 AC 90 00016	MOVW	COND, 4(R0)	: 2711
06	A0	04		04 0001B	MOVB	LIT, 6(R0)	: 2714
					RET		:

; Routine Size: 28 bytes, Routine Base: \$CODE\$ + 137C

SRTTRN
V04-000

M 3
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 103
(33)

SR
VO

```
: 2651      2715 1 ROUTINE PUT_QUOT (LEN, ADR: REF VECTOR[,BYTE]): NOVALUE =
: 2652      2716 2 BEGIN
: 2653      2717 2 LOCAL
: 2654      2718 2     BUF: VECTOR[OUT_BUF_WIDTH, BYTE],
: 2655      2719 2     PTR1: REF VECTOR[,BYTE],
: 2656      2720 2     PTR2: REF VECTOR[,BYTE];
: 2657      2721 2     PTR1 = ADR[0];
: 2658      2722 2     PTR2 = BUF[0];
: 2659      2723 2     CH$WCHAR A(%C'', PTR2);
: 2660      2724 2     DECR I FROM .LEN-1 TO 0 DO
: 2661      2725 2     BEGIN
: 2662      2726 2     CH$WCHAR A( CH$RCHAR(.PTR1), PTR2 );
: 2663      2727 2     IF CH$RCHAR_A(PTR1) EQL %C'' THEN CH$WCHAR_A(%C'', PTR2);
: 2664      2728 2     END;
: 2665      2729 2     CH$WCHAR A(%C'', PTR2);
: 2666      2730 2     PUT_TEXT( CH$DIFF(.PTR2, BUF[0]), BUF[0] );
: 2667      2731 1     END;
```

000C 00000 PUT_QUOT:						
				.WORD	Save R2,R3	: 2715
5E	FF00	CE	9E 00002	MOVAB	-256(SP), SP	
53	08	AC	D0 00007	MOVL	ADR, PTR1	: 2721
51		6E	9E 0000B	MOVAB	BUF, PTR2	: 2722
81		22	90 0000E	MOVB	#34, (PTR2)+	: 2723
50	04	AC	D0 00011	MOVL	LEN, I	: 2724
		0E	11 00015	BRB	2\$	
81		63	90 00017 1\$:	MOVB	(PTR1), (PTR2)+	: 2726
52		83	9A 0001A	MOVZBL	(PTR1)+, R2	: 2727
22		52	91 0001D	CMPB	R2, #34	
		03	12 00020	BNEQ	2\$	
81		22	90 00022	MOVB	#34, (PTR2)+	
EF		50	F4 00025 2\$:	SOBGEQ	I, 1\$	: 2724
81		22	90 00028	MOVB	#34, (PTR2)+	: 2729
		5E	DD 0002B	PUSHL	SP	: 2730
7E		AE	9E 0002D	MOVAB	BUF, R0	
		50	C3 00031	SUBL3	R0, PTR2, -(SP)	
0000V	CF	02	FB 00035	CALLS	#2, PUT_TEXT	
		04	0003A	RET		: 2731

; Routine Size: 59 bytes, Routine Base: \$CODE\$ + 1398


```
: 2669      2732 1 ROUTINE PUT_HEX (LEN, ADR: REF VECTOR[BYTE]): NOVALUE =
: 2670      2733 2   BEGIN
: 2671      2734 2   LOCAL
: 2672      2735 2   BUF: VECTOR[18, BYTE];
: 2673      2736 2   PTR: REF VECTOR[BYTE];
: 2674      2737 2   PTR = BUF[0];
: 2675      2738 2   CH$WCHAR_A(%C'%', PTR);
: 2676      2739 2   CH$WCHAR_A(%C'X', PTR);
: 2677      2740 2   DECF I FROM .LEN-1 TO 0 DO
: 2678      2741 3   BEGIN
: 2679      2742 3   LOCAL R: BYTE;
: 2680      2743 3   BIND X = UPLIT BYTE('0123456789ABCDEF'): VECTOR[BYTE];
: 2681      2744 3   R = .ADR[I];
: 2682      2745 3   CH$WCHAR_A(.X[R<4,4,0>], PTR);
: 2683      2746 3   CH$WCHAR_A(.X[R<0,4,0>], PTR);
: 2684      2747 2   END;
: 2685      2748 2   PUT_TEXT( CH$DIFF(.PTR, BUF[0]), BUF[0] );
: 2686      2749 1   END;
```

```
.PSECT $PLITS$,NOWRT,NOEXE,2
45 44 43 42 41 39 38 37 36 35 34 33 32 31 30 002F9 P.ADN: .ASCII \0123456789ABCDEF\
46 00308
```

X=

P.ADN

```
.PSECT $CODE$,NOWRT,2
000C 00000 PUT_HEX: .WORD Save R2,R3
5E      14 C2 00002   SUBL2 #20, SP
52      6E 9E 00005   MOVAB BUF, PTR
82      5825 8F B0 00008   MOVW #22565, (PTR)+
51      04 AC D0 0000D   MOVL LEN, I
53      08 BC41 90 00011   BRB 2$
50      04 04 EF 00018   1$: MOVB @ADR[I], R
50      04 0000'CF40 90 0001D   EXTZV #4, #4, R, R0
82      0000'CF40 90 0001D   MOVAB X[R0], (PTR)+
50      04 00 EF 00023   EXTZV #0, #4, R, R0
82      0000'CF40 90 00028   MOVAB X[R0], (PTR)+
E2      51 F4 0002E   2$: SOBGEQ I, 1$
50      5E DD 00031   PUSHL SP
7E      04 AE 9E 00033   MOVAB BUF, R0
52      50 C3 00037   SUBL3 R0, PTR, -(SP)
0000V CF 02 FB 0003B   CALLS #2, PUT_TEXT
04 00040   RET
```

; Routine Size: 65 bytes, Routine Base: \$CODE\$ + 13D3


```
2688 2750 1 ROUTINE PUT_NUMB (LEN, ADR: REF VECTOR[,BYTE], NAM: REF FLD_BLOCK): NOVALUE =
2689 2751 2 BEGIN
2690 2752 2 LOCAL
2691 2753 2 SIZ;
2692 2754 2
2693 2755 2 ! Determine how large a field is needed.
2694 2756 2 !
2695 2757 2 SIZ = .NAM[FLD_SIZ];
2696 2758 2 !<s1>
2697 2759 2 SELECT ONE .NAM[FLD_DTY] OF
2698 2760 2 SET
2699 2761 2 [DSC$K-DTYPE-B,
2700 2762 2 DSC$K-DTYPE-BU]:
2701 2763 2 BEGIN
2702 2764 2 LOCAL
2703 2765 2 RES: INITIAL(0);
2704 2766 2 IF NOT CVT_DTB(.LEN, ADR[0], RES) THEN SOR_ERROR(SRTTRN$_INV_CONST);
2705 2767 2 PUT_HEX( MINU(.SIZ,%ALLOCATION(RES)), RES );
2706 2768 2 RETURN;
2707 2769 2 END;
2708 2770 2 [DSC$K-DTYPE-NL,
2709 2771 2 DSC$K-DTYPE-NR]: SIZ = .SIZ + 1;
2710 2772 2 [DSC$K-DTYPE-P]: SIZ = .SIZ / 2 + 1;
2711 2773 2 [OTHERWISE]: 0;
2712 2774 2 TES;
2713 2775 2 PUT_QUOT(MINU(.LEN,.SIZ), ADR[0]);
2714 2776 1 END;
```

0004 00000 PUT_NUMB:						
	5E		04 C2 00002	.WORD	Save R2	2750
	50	0C	AC D0 00005	SUBL2	#4, SP	2757
	52	0A	A0 3C 00009	MOVL	NAM, R0	
	50	10	A0 9A 0000D	MOVZWL	10(R0), SIZ	2759
	02		50 91 00011	MOVZBL	16(R0), R0	2761
			05 13 00014	CMPB	R0, #2	
	06		50 91 00016	BEQL	1\$	
			2D 12 00019	CMPB	R0, #6	
			6E D4 0001B 1\$:	BNEQ	4\$	
			5E DD 0001D	CLRL	RES	2763
	7E	04	AC 7D 0001F	PUSHL	SP	2766
0000V	CF		03 FB 00023	MOVQ	LEN, -(SP)	
	0B		50 E8 00028	CALLS	#3, CVT_DTB	
		001C800C	8F DD 0002B	BLBS	R0, 2\$	
EBB6	CF		01 FB 00031	PUSHL	#1867788	
		4004	8F BB 00036 2\$:	CALLS	#1, SOR_ERROR	
	04		6E D1 0003A	PUSHR	#^M<R2,SP>	2767
			03 1B 0003D	CMPB	(SP), #4	
	6E		04 D0 0003F	BLEQU	3\$	
FF78	CF		02 FB 00042 3\$:	MOVL	#4, (SP)	
			04 00047	CALLS	#2, PUT_HEX	
	10		50 91 00048 4\$:	RET		2763
			05 13 0004B	CMPB	R0, #16	2770
				BEQL	5\$	

SRTTRN
V04-000

D 4
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 107
(36)

	12		50	91	0004D	CMPB	R0, #18		
			04	12	00050	BNEQ	6\$		
			52	D6	00052	INCL	SIZ	2771	
			0D	11	00054	BRB	7\$		
	15		50	91	00056	CMPB	R0, #21	2772	
			08	12	00059	BNEQ	7\$		
50	52		02	C7	0005B	DIVL3	#2, SIZ, R0		
	52	01	A0	9E	0005F	MOVAB	1(R0), SIZ		
	7E	04	AC	7D	00063	MOVQ	LEN, -(SP)	2775	
	52		6E	D1	00067	CMPL	(SP), SIZ		
			03	1B	0006A	BLEQU	8\$		
	6E		52	D0	0006C	MOVL	SIZ, (SP)		
FF10	CF		02	FB	0006F	CALLS	#2, PUT_QUOT		
			04	00	00074	RET		2776	

; Routine Size: 117 bytes, Routine Base: \$CODE\$ + 1414

SRT
V04

```
: 2716      2777 1 ROUTINE PUT_FLOT (LEN, ADR: REF VECTOR[,BYTE], ITMLEN): NOVALUE =
: 2717      2778 2 BEGIN
: 2718      2779 2 LOCAL
: 2719      2780 2 PTR: REF VECTOR[,BYTE],
: 2720      2781 2 DESC: VECTOR[2],
: 2721      2782 2 RESULT: VECTOR[8,BYTE],
: 2722      2783 2 STATUS;
: 2723      2784 2
: 2724      2785 2 ! Find the end of the string, and convert the text string to D-floating.
: 2725      2786 2 !
: 2726      2787 2 PTR = CH$FIND.CH(.LEN, ADR[0], %C' ');
: 2727      2788 2 IF CH$FAIL(.PTR) THEN PTR = ADR[.LEN];
: 2728      2789 2
: 2729      2790 2 DESC[0] = CH$DIFF(.PTR, ADR[0]);
: 2730      2791 2 DESC[1] = ADR[0];
: 2731      2792 2
: 2732      2793 2 ! Convert the number to floating
: 2733      2794 2 !
: 2734      2795 2 BEGIN
: 2735      2796 2 EXTERNAL ROUTINE OTSS$CVT_T_D;
: 2736      2797 2 STATUS = OTSS$CVT_T_D(DESC[0], RESULT[0]);
: 2737      2798 2 IF NOT .STATUS
: 2738      2799 2 THEN
: 2739      2800 2 SOR_ERROR(SRTTRN$_INV_CONST AND NOT STSSM_SEVERITY OR STSSK_ERROR);
: 2740      2801 2 END;
: 2741      2802 2
: 2742      2803 2 ! Now output the number, in HEX
: 2743      2804 2 !
: 2744      2805 2 PUT_HEX (.ITMLEN, RESULT[0]);
: 2745      2806 2
: 2746      2807 1 END;
```

```
.EXTRN OTSS$CVT_T_D
0000 0000 PUT_FLOT:
08 BC 04 5E 10 C2 00002 .WORD Save nothing : 2777
AC 20 3A 00005 SUBL2 #16, SP : 2787
02 12 0000B LOCC #32, LEN, @ADR
51 D4 0000D BNEQ 1$
51 D5 0000F 1$: TSTL PTR : 2788
06 12 00011 BNEQ 2$
08 51 08 AC 04 AC C1 00013 ADDL3 LEN, ADR, PTR : 2790
AE 51 08 AC 08 AC C3 00019 2$: SUBL3 ADR, PTR, DESC : 2791
OC AE 08 AC 08 AC D0 0001F MOVL ADR, DESC+4 : 2797
OC AE 5E DD 00024 PUSHL SP
OC AE 9F 00026 PUSHAB DESC
00000000G 00 02 FB 00029 CALLS #2, OTSS$CVT_T_D : 2798
OB 50 E8 00030 BLBS STATUS, 3$ : 2800
EB39 CF 001C800A 8F DD 00033 PUSHL #1867786
01 FB 00039 CALLS #1, SOR_ERROR : 2805
5E DD 0003E 3$: PUSHL SP
OC AC DD 00040 PUSHL ITMLEN
FF02 CF 02 FB 00043 CALLS #2, PUT_HEX : 2807
04 00048 RET
```


SRTTRN
V04-000

F 4
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 109
(37)

; Routine Size: 73 bytes, Routine Base: \$CODE\$ + 1489

```
: 2748 2808 1 ROUTINE PUT_LITE (LEN, ADR: REF VECTOR[BYTE], NAM: REF FLD_BLOCK): NOVALUE =
: 2749 2809 BEGIN
: 2750 2810 !<s1>
: 2751 2811 SELECTONE .NAM[FLD_DTY] OF
: 2752 2812 SET
: 2753 2813 [DSC$K_DTYPE_B,
: 2754 2814 DSC$K_DTYPE_BU]: PUT_NUMB(.LEN, ADR[0], NAM[BASE_]);
: 2755 2815 [DSC$K_DTYPE_T]: PUT_QUOT(MINU(.LEN,.NAM[FLD_SIZ]), ADR[0]);
: 2756 2816 [DSC$K_DTYPE_NRO,
: 2757 2817 DSC$K_DTYPE_NL,
: 2758 2818 DSC$K_DTYPE_NR,
: 2759 2819 DSC$K_DTYPE_NLO,
: 2760 2820 DSC$K_DTYPE_NU,
: 2761 2821 DSC$K_DTYPE_P,
: 2762 2822 DSC$K_DTYPE_DIBOL,
: 2763 2823 DSC$K_DTYPE_AZ,
: 2764 2824 DSC$K_DTYPE_NZ]: PUT_NUMB(.LEN, ADR[0], NAM[BASE_]);
: 2765 2825 [DSC$K_DTYPE_F]: PUT_FLOT(.LEN, ADR[0], 4);
: 2766 2826 [DSC$K_DTYPE_D]: PUT_FLOT(.LEN, ADR[0], 8);
: 2767 2827 [OTHERWISE]: RETURN SOR_ERROR(SRTTRN$_INV_DATA);
: 2768 2828 TES;
: 2769 2829 1 END;
```

000C 00000 PUT_LITE:						
	53	0C	AC D0 00002	.WORD	Save R2,R3	2808
	52	10	A3 9A 00006	MOVL	NAM, R3	2811
	02		52 91 0000A	MOVZBL	16(R3), R2	
			34 13 0000D	CMPB	R2, #2	2813
	06		52 91 0000F	BEQL	4\$	
			2F 13 00012	CMPB	R2, #6	
	0E		52 91 00014	BEQL	4\$	
			16 12 00017	CMPB	R2, #14	2815
	7E	04	AC 7D 00019	BNEQ	2\$	
6E	10		00 ED 0001D	MOVQ	LEN, -(SP)	
			04 1E 00023	CMPZV	#0, #16, 10(R3), (SP)	
	6E	0A	A3 3C 00025	BGEQU	1\$	
FE98	CF		02 FB 00029 1\$:	MOVZWL	10(R3), (SP)	
			04 0002E	CALLS	#2, PUT_QUOT	
	0F		52 91 0002F 2\$:	RET		
			05 1F 00032	CMPB	R2, #15	2816
	15		52 91 00034	BLSSU	3\$	
			0A 1B 00037	CMPB	R2, #21	
	28		52 91 00039 3\$:	BLEQU	4\$	
			11 1F 0003C	CMPB	R2, #40	
	29		52 91 0003E	BLSSU	5\$	
			0C 1A 00041	CMPB	R2, #41	
			53 DD 00043 4\$:	BGTRU	5\$	
	7E	04	AC 7D 00045	PUSHL	R3	2824
FEF4	CF		03 FB 00049	MOVQ	LEN, -(SP)	
			04 0004E	CALLS	#3, PUT_NUMB	
			52 91 0004F 5\$:	RET		
	0A		04 12 00052	CMPB	R2, #10	2825
				BNEQ	6\$	

SRTTRN
V04-000

H 4
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 111
(38)

			04	DD	00054		PUSHL	#4		
			07	11	00056		BRB	7\$		
	0B		52	91	00058	6\$:	CMPB	R2, #11		2826
			0C	12	0005B		BNEQ	8\$		
			08	DD	0005D		PUSHL	#8		
	7E	04	AC	7D	0005F	7\$:	MOVQ	LEN, -(SP)		
FF4F	CF		03	FB	00063		CALLS	#3, PUT_FLOT		
				04	00068		RET			
			8F	DD	00069	8\$:	PUSHL	#1867804		2827
EABA	CF	001C801C	01	FB	0006F		CALLS	#1, SOR_ERROR		
			04	00074			RET			2829

; Routine Size: 117 bytes, Routine Base: \$CODE\$ + 14D2

SRT
V04

```
: 2771 2830 1 ROUTINE PUT_TEXT (LEN, ADR): NOVALUE CALL =
: 2772 2831 1
: 2773 2832 1 !++
: 2774 2833 1
: 2775 2834 1 FUNCTIONAL DESCRIPTION:
: 2776 2835 1
: 2777 2836 1 Put some text to the output buffer.
: 2778 2837 1
: 2779 2838 1 FORMAL PARAMETERS:
: 2780 2839 1
: 2781 2840 1 LEN Length of the text
: 2782 2841 1 ADR Address of the text
: 2783 2842 1
: 2784 2843 1 IMPLICIT INPUTS:
: 2785 2844 1
: 2786 2845 1 NONE
: 2787 2846 1
: 2788 2847 1 IMPLICIT OUTPUTS:
: 2789 2848 1
: 2790 2849 1 NONE
: 2791 2850 1
: 2792 2851 1 ROUTINE VALUE:
: 2793 2852 1
: 2794 2853 1 0 or 1
: 2795 2854 1
: 2796 2855 1 SIDE EFFECTS:
: 2797 2856 1
: 2798 2857 1 NONE
: 2799 2858 1 --
: 2800 2859 2 BEGIN
: 2801 2860 2 BUILTIN
: 2802 2861 2 ACTUALCOUNT,
: 2803 2862 2 ACTUALPARAMETER;
: 2804 2863 2 MACRO
: 2805 2864 2 LEN_(X) = ACTUALPARAMETER(X) %,
: 2806 2865 2 ADR_(X) = ACTUALPARAMETER(X+1) %;
: 2807 2866 2 LOCAL
: 2808 2867 2 TOT_LEN;
: 2809 2868 2
: 2810 2869 2 IF .LEN GTR 0 THEN
: 2811 2870 2 IF CH$RCHAR(.ADR) EQL %C'/'
: 2812 2871 2 THEN
: 2813 2872 2 PUT_LINE();
: 2814 2873 2
: 2815 2874 2 TOT_LEN = 0;
: 2816 2875 2 INCR I FROM 1 TO ACTUALCOUNT()-1 BY 2 DO
: 2817 2876 2 TOT_LEN = .TOT_LEN + LEN_(.I);
: 2818 2877 2
: 2819 2878 2 IF .TOT_LEN GTRU .OUT_DSC[0] AND .TOT_LEN LSSU OUT_WIDTH
: 2820 2879 2 THEN
: 2821 2880 2 PUT_LINE(1);
: 2822 2881 2
: 2823 2882 2 INCR I FROM 1 TO ACTUALCOUNT()-1 BY 2 DO
: 2824 2883 2 BEGIN
: 2825 2884 2 OUT_DSC[1] = CH$MOVE(LEN_(.I), ADR_(.I), .OUT_DSC[1]);
: 2826 2885 2 OUT_DSC[0] = .OUT_DSC[0] - LEN_(.I);
: 2827 2886 2 END;
```


: 2828
: 2829

2887 2
2888 1

END;

		01FC 00000 PUT_TEXT:				
	58	0000'	CF 9E 00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8	: 2830
		04	AC D5 00007	MOVAB	OUT_DSC, R8	: 2869
			0B 15 0000A	TSTL	LEN	: 2870
	2F	08	BC 91 0000C	BLEQ	1\$	: 2872
			05 12 00010	CMPB	@ADR, #47	: 2874
	0000V	CF	00 FB 00012	BNEQ	1\$	: 2875
			52 D4 00017	CALLS	#0, PUT_LINE	: 2876
	51		6C 9A 00019	CLRL	TOT_LEN	: 2878
			51 D7 0001C	MOVZBL	(AP), R1	: 2880
	50		01 CE 0001E	DECL	R1	: 2882
			04 11 00021	MNEGL	#1, I	: 2884
	52	6C40	C0 00023	BRB	3\$	: 2885
FFF6	50		51 F1 00027	ADDL2	(AP)[I], TOT_LEN	: 2886
	02		52 D1 0002D	ACBL	R1, #2, I, 2\$	: 2887
	68		10 1B 00030	CMP	TOT_LEN, OUT_DSC	: 2888
			52 D1 00032	BLEQU	4\$	: 2889
	00000047	8F	07 1E 00039	CMP	TOT_LEN, #71	: 2890
			01 DD 0003B	BGEQU	4\$	: 2891
	0000V	CF	01 FB 0003D	PUSHL	#1	: 2892
			57 6C 9A 00042	CALLS	#1, PUT_LINE	: 2893
	56		57 D7 00045	MOVZBL	(AP), R7	: 2894
			01 CE 00047	DECL	R7	: 2895
	50		15 11 0004A	MNEGL	#1, I	: 2896
			04 AC46 D0 0004C	BRB	6\$	: 2897
			6C46 DF 00051	MOVL	4(AP)[I], R0	: 2898
	04	88	9E 28 00054	PUSHAL	(AP)[I]	: 2899
			53 D0 00059	MOV3	@(SP)+, (R0), @OUT_DSC+4	: 2900
	04	68	6C46 C2 0005D	MOVL	R3, OUT_DSC+4	: 2901
FFE5	56		57 F1 00061	SUBL2	(AP)[I], OUT_DSC	: 2902
			04 00067	ACBL	R7, #2, I, 5\$	: 2903
				RET		: 2904

; Routine Size: 104 bytes, Routine Base: \$CODE\$ + 1547

```

: 2831 2889 1 ROUTINE PUT_LINE(TAB): NOVALUE CALL =
: 2832 2890 1
: 2833 2891 1 !++
: 2834 2892 1
: 2835 2893 1 FUNCTIONAL DESCRIPTION:
: 2836 2894 1
: 2837 2895 1     Output a line
: 2838 2896 1
: 2839 2897 1 FORMAL PARAMETERS:
: 2840 2898 1
: 2841 2899 1     TAB     Amount to tab over on the new line (optional)
: 2842 2900 1
: 2843 2901 1 IMPLICIT INPUTS:
: 2844 2902 1
: 2845 2903 1     NONE
: 2846 2904 1
: 2847 2905 1 IMPLICIT OUTPUTS:
: 2848 2906 1
: 2849 2907 1     NONE
: 2850 2908 1
: 2851 2909 1 ROUTINE VALUE:
: 2852 2910 1
: 2853 2911 1     0 or 1
: 2854 2912 1
: 2855 2913 1 SIDE EFFECTS:
: 2856 2914 1
: 2857 2915 1     NONE
: 2858 2916 1 --
: 2859 2917 2 BEGIN
: 2860 2918 2 BUILTIN
: 2861 2919 2 NULLPARAMETER;
: 2862 2920 2
: 2863 2921 2 ! Write the line to the file
: 2864 2922 2 !
: 2865 2923 2 OUT_RAB[RAB$W_RSZ] = .OUT_DSC[1] - OUT_BUF[0];
: 2866 2924 2
: 2867 2925 2 OUT_RAB[RAB$L_CTX] = SRTTRN$ WRITEERR;
: 2868 2926 2 $PUT( RAB=OUT_RAB[BASE_], ERR=REPORT_IO_ERROR );
: 2869 2927 2
: 2870 2928 2 OUT_DSC[0] = OUT_WIDTH;
: 2871 2929 2 OUT_DSC[1] = OUT_BUF[0];
: 2872 2930 2
: 2873 2931 2 IF NOT NULLPARAMETER(1)
: 2874 2932 2 THEN
: 2875 2933 2     PUT_TEXT(.TAB, UPLIT BYTE(REP 10 OF (%CHAR(9)))));
: 2876 2934 2
: 2877 2935 1 END;

```

```

                                .PSECT $PLITS,NOWRT,NOEXE,2
09 00309 P.ADO: .ASCII <9>
09 0030A       .ASCII <9>
09 0030B       .ASCII <9>
09 0030C       .ASCII <9>
09 0030D       .ASCII <9>

```

...


```
09 0030E .ASCII <9>
09 0030F .ASCII <9>
09 00310 .ASCII <9>
09 00311 .ASCII <9>
09 00312 .ASCII <9>
```

```
.EXTRN  SYSS$PUT
```

```
.PSECT $CODE$,NOWRT,2
```

```
0000 00000 PUT_LINE:
```

Address	Op Code	Op Name	Comment	PC
0000	CF	50	0000	CF 9E 00002
		CF	50	A3 00007
	0000	CF	001C10D2	8F D0 0000F
			EC1C	CF 9F 00018
		0000		CF 9F 0001C
00000000G	00		02	FB 00020
0000	CF		47	8F 9A 00027
0000	CF		0000	CF 9E 0002D
				6C 95 00034
				11 13 00036
		04		AC D5 00038
				0C 13 0003B
		0000		CF 9F 0003D
		04		AC DD 00041
FF4F	CF		02	FB 00044
			04	00049 1\$:
				WORD
				MOVAB OUT_BUF, R0
				SUBW3 R0, OUT_DSC+4, OUT_RAB+34
				MOVL #1839314, OUT_RAB+24
				PUSHAB REPORT_IO_ERROR
				PUSHAB OUT_RAB
				CALLS #2, SYSS\$PUT
				MOVZBL #71, OUT_DSC
				MOVAB OUT_BUF, OUT_DSC+4
				TSTB (AP)
				BEQL 1\$
				TSTL 4(AP)
				BEQL 1\$
				PUSHAB P.ADO
				PUSHL TAB
				CALLS #2, PUT_TEXT
				RET

; Routine Size: 74 bytes, Routine Base: \$CODES + 15AF

```
2879 2936 1 ROUTINE CLOSE_FILES : NOVALUE =
2880 2937 1
2881 2938 1 ++
2882 2939 1
2883 2940 1 FUNCTIONAL DESCRIPTION:
2884 2941 1
2885 2942 1     Close the input and output files.
2886 2943 1
2887 2944 1 FORMAL PARAMETERS:
2888 2945 1
2889 2946 1     NONE
2890 2947 1
2891 2948 1 IMPLICIT INPUTS:
2892 2949 1
2893 2950 1     NONE
2894 2951 1
2895 2952 1 IMPLICIT OUTPUTS:
2896 2953 1
2897 2954 1     NONE
2898 2955 1
2899 2956 1 ROUTINE VALUE:
2900 2957 1
2901 2958 1     0 or 1
2902 2959 1
2903 2960 1 SIDE EFFECTS:
2904 2961 1
2905 2962 1     NONE
2906 2963 1 --
2907 2964 2 BEGIN
2908 2965 2
2909 2966 2     ! Close input file
2910 2967 2
2911 2968 2     INP_FAB[FAB$L CTX] = SRTTRN$ CLOSEIN;
2912 2969 2     $CLOSE( FAB=INP_FAB[BASE_], ERR=REPORT_IO_ERROR );
2913 2970 2
2914 2971 2     ! Close temporary file
2915 2972 2
2916 2973 2     IF .TMP_FAB[FAB$W_IFI] NEQ 0
2917 2974 2     THEN
2918 2975 2         BEGIN
2919 2976 2             TMP_FAB[FAB$L CTX] = SRTTRN$ CLOSEDEL;
2920 2977 2             $CLOSE( FAB=TMP_FAB[BASE_], ERR=REPORT_IO_ERROR );
2921 2978 2             END;
2922 2979 2
2923 2980 2     ! Close output file
2924 2981 2
2925 2982 2     OUT_FAB[FAB$L CTX] = SRTTRN$ CLOSEOUT;
2926 2983 2     $CLOSE( FAB=OUT_FAB[BASE_], ERR=REPORT_IO_ERROR );
2927 2984 2
2928 2985 1 END;
```

.EXTRN SYSS\$CLOSE

000C 00000 CLOSE_FILES:

.WORD Save R2,R3

: 2936

SRTTRN
V04-000

N 4
16-Sep-1984 01:11:52 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:54 [SORT32.SRC]SRTTRN.B32;1

Page 117
(41)

	53	EBE8	CF	9E	00002	MOVAB	REPORT_IO_ERROR, R3	
	52	00000000G	00	9E	00007	MOVAB	SYSS\$CLOSE, R2	
0000'	CF	001C1052	8F	D0	0000E	MOVL	#1839186, INP_FAB+24	2968
			53	DD	00017	PUSHL	R3	2969
		0000'	CF	9F	00019	PUSHAB	INP_FAB	
62			02	FB	0001D	CALLS	#2, -SYSS\$CLOSE	
		0000'	CF	B5	00020	TSTW	TMP_FAB+2	2973
			12	13	00024	BEQL	1\$	
0000'	CF	001C121A	8F	D0	00026	MOVL	#1839642, TMP_FAB+24	2976
			53	DD	0002F	PUSHL	R3	2977
		0000'	CF	9F	00031	PUSHAB	TMP_FAB	
62			02	FB	00035	CALLS	#2, -SYSS\$CLOSE	
0000'	CF	001C105A	8F	D0	00038	MOVL	#1839194, OUT_FAB+24	2982
			53	DD	00041	PUSHL	R3	2983
		0000'	CF	9F	00043	PUSHAB	OUT_FAB	
62			02	FB	00047	CALLS	#2, -SYSS\$CLOSE	
			04	00	0004A	RET		2985

; Routine Size: 75 bytes, Routine Base: \$CODE\$ + 15F9


```
2930 2986 1 ROUTINE GET_LINE =
2931 2987 1
2932 2988 1 ++
2933 2989 1
2934 2990 1 FUNCTIONAL DESCRIPTION:
2935 2991 1
2936 2992 1     Get a line from the input file.
2937 2993 1
2938 2994 1 FORMAL PARAMETERS:
2939 2995 1
2940 2996 1     NONE
2941 2997 1
2942 2998 1 IMPLICIT INPUTS:
2943 2999 1
2944 3000 1     NONE
2945 3001 1
2946 3002 1 IMPLICIT OUTPUTS:
2947 3003 1
2948 3004 1     NONE
2949 3005 1
2950 3006 1 ROUTINE VALUE:
2951 3007 1
2952 3008 1     0 or 1
2953 3009 1
2954 3010 1 SIDE EFFECTS:
2955 3011 1
2956 3012 1     NONE
2957 3013 1 --
2958 3014 2 BEGIN
2959 3015 2 LOCAL
2960 3016 2 STATUS;
2961 3017 2
2962 3018 2 DO
2963 3019 2 BEGIN
2964 3020 2
2965 3021 2 CUR_RAB[RAB$L_CTX] = SRTTRN$ READERR;
2966 3022 2 STATUS = $GET( RAB=CUR_RAB[BASE_], ERR=REPORT_IO_ERROR );
2967 3023 2
2968 3024 2 ! Check for end of file
2969 3025 2 !
2970 3026 2 IF .STATUS EQL RMSS_EOF
2971 3027 2 THEN
2972 3028 2 BEGIN
2973 3029 2 LINE_TYPE = 0;
2974 3030 2 RETURN FALSE;
2975 3031 2 END;
2976 3032 2
2977 3033 2 ! Get record size from record access block
2978 3034 2
2979 3035 2 INP_REC_SIZ = .CUR_RAB[RAB$W_RSZ];
2980 3036 2
2981 3037 2 ! Fill unused portion of buffer with blanks
2982 3038 2 !
2983 3039 2 CH$FILL(%C' ',
2984 3040 2 %ALLOCATION(INP_BUF)-.INP_REC_SIZ,
2985 3041 2 INP_BUF[.INP_REC_SIZ]);
2986 3042 2
```



```
2987 3043 3      ! If this is the first pass, output the line as a comment
2988 3044 3      !
2989 3045 3      IF .PASS EQL 1
2990 3046 3      THEN
2991 3047 4          BEGIN
2992 3048 4              !
2993 3049 4              ! Issue the line as a comment.
2994 3050 4              ! If the input file is not on a rewindable device,
2995 3051 4              ! also write the record to the temporary file.
2996 3052 4              !
2997 3053 4              PUT_TEXT(
2998 3054 4                  LENADR('!'),
2999 3055 4                  .INP_REC_SIZ, INP_BUF[0]);
3000 3056 4              PUT_LINE();
3001 3057 4              IF .TMP_FAB[FAB$W_IFI] NEQ 0
3002 3058 4              THEN
3003 3059 5                  BEGIN
3004 3060 5                      TMP_RAB[RAB$W_RSZ] = .CUR_RAB[RAB$W_RSZ];
3005 3061 5                      TMP_RAB[RAB$L_CTX] = SRTTRN$ WRITEERR;
3006 3062 5                      $PUT( RAB=TMP_RAB[BASE_], ERR=REPORT_IO_ERROR );
3007 3063 4                      END;
3008 3064 3              END;
3009 3065 3      IF .INP_BUF[6] EQL %C'*'
3010 3066 3      THEN
3011 3067 3          LINE_TYPE = 0
3012 3068 3      ELIF
3013 3069 3          CH$EQL(.INP_REC_SIZ, INP_BUF[0], 0, INP_BUF[0], %C' ')
3014 3070 3      THEN
3015 3071 3          LINE_TYPE = 0
3016 3072 3      ELSE
3017 3073 3          BEGIN
3018 3074 4              LOCAL
3019 3075 4                  TEMP;
3020 3076 4              !
3021 3077 4              ! Convert first 6 character positions of record
3022 3078 4              !
3023 3079 4              UPCASE(6, INP_BUF[0]);
3024 3080 4              !
3025 3081 4              ! Find type of record
3026 3082 4              !
3027 3083 4              IF CH$EQL(LENADR('ALTSEQ'),6,INP_BUF[0],%C' ')
3028 3084 4              THEN
3029 3085 4                  BEGIN
3030 3086 5                      LINE_TYPE = ALTSEQ_LINE;
3031 3087 5                      RETURN TRUE;
3032 3088 5                  END;
3033 3089 4              !
3034 3090 4              ! Check whether this record is in free format, and convert to
3035 3091 4              ! fixed format if it is.
3036 3092 4              !
3037 3093 4              FREE_FIXED();
3038 3094 4              !
3039 3095 4              ! Check line number order (if not altseq)
3040 3096 4              !
3041 3097 4              IF .PASS EQL 1 AND NOT .LINE_SEQ
3042 3098 4              THEN
3043 3099 4
```



```
3044 BEGIN
3045 IF CH$EQL(5, INP_BUF[0], 0, INP_BUF[0], %C' ')
3046 THEN
3047     0 ! Ignore blank line numbers
3048 ELSE
3049     BEGIN
3050     IF CVT_DTB(5, INP_BUF[0], TEMP)
3051     THEN .TEMP GTR .LINE_NUM
3052     ELSE FALSE
3053     END
3054 THEN
3055     LINE_NUM = .TEMP
3056 ELSE
3057     BEGIN
3058     SOR_ERROR(SRTTRN$_LINE_NUM);
3059     LINE_SEQ = TRUE;
3060     END;
3061 END;
3062 ! Determine the type of line
3063 !
3064 LINE_TYPE = (SELECTONE .INP_BUF[5] OF
3065 SET
3066 ['H']: HEADER_LINE;
3067 ['I']: RECORD_LINE;
3068 ['O']: RECORD_LINE;
3069 ['F']: FIELD_LINE;
3070 [OTHERWISE]:
3071     (IF .PASS EQL 1 THEN SOR_ERROR(SRTTRN$_INV_TYPE);0);
3072 TES);
3073 END;
3074
3075 END
3076 UNTIL .LINE_TYPE NEQ 0;
3077
3078 ! If the user is translating an old-format Sort-32 spec file,
3079 ! apply some defaults for things that Sort-32 never implemented.
3080 IF .SWITCH_S32 AND .LINE_TYPE EQL HEADER_LINE
3081 THEN
3082     BEGIN
3083     INP_BUF[25] = %C' ': ! Ascii collating sequence
3084     INP_BUF[27] = %C'X': ! Strip the keys
3085     END;
3086
3087 RETURN TRUE;
3088 END;
```

.PSECT \$PLITS,NOWRT,NOEXE,2

51 45 53 54 4C 21 00313 P.ADP: .ASCII \\
41 00314 P.ADQ: .ASCII \\ALTSEQ\

.EXTRN SYSSGET


```
.PSECT $CODE$,NOWRT,2

03FC 00000 GET_LINE:
      59      0000'  CF  9E 00002      .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9      : 2986
      58      0000'  CF  9E 00007      MOVAB      LINE_TYPE, R9
      5E      04  C2 0000C      MOVAB      INP_BUF, R8
      50      0190  C8  D0 0000F  1$:      SUBL2      #4, SP
      18      A0 001C10B2  8F  D0 00014      MOVL      CUR_RAB, R0      : 3021
      EB83      CF  9F 0001C      MOVL      #1839282, 24(R0)
      50      DD 00020      PUSHAB     REPORT_IO_ERROR      : 3022
      00000000G  00      02  FB 00022      PUSHL      R0
      57      50  D0 00029      CALLS      #2, SYSS$GET
      0001827A  8F      57  D1 0002C      MOVL      R0, STATUS
      05      12 00033      CMPL      STATUS, #98938      : 3026
      69      D4 00035      BNEQ      2$
      012D      31 00037      CLRL      LINE_TYPE
      50      0190  C8  D0 0003A  2$:      BRW      18$
      018C      22  A0 3C 0003F      MOVL      CUR_RAB, R0
      56      018C  C8  D0 00045      MOVZWL     34(R0), INP_REC_SIZ
      50 00000084  8F      56  C3 0004A      MOVL      INP_REC_SIZ, R6
      20      6E      00  2C 00052      SUBL3      R6, #132, R0
      6846      00057      MOVCS      #0, (SP), #32, R0, INP_BUF[R6]
      01      14  A9  D1 00059      CMPL      PASS, #1
      3D      12 0005D      BNEQ      3$
      0140      8F  BB 0005F      PUSHR      #^M<R6,R8>
      0000'      CF  9F 00063      PUSHAB     P.ADP
      FE95      CF      01  DD 00067      PUSHL      #1
      FEF8      CF      04  FB 00069      CALLS      #4, PUT_TEXT
      04EA      00  FB 0006E      CALLS      #0, PUT_LINE
      23      B5 00073      TSTW      TMP_FAB+2
      50      0190  C8  D0 00079      BEQL      3$
      055A      22  A0  B0 0007E      MOVL      CUR_RAB, R0
      0550      C8 001C10D2  8F  D0 00084      MOVW      34(R0), TMP_RAB+34
      EB12      CF  9F 0008D      MOVL      #1839314, TMP_RAB+24
      0538      C8  9F 00091      PUSHAB     REPORT_IO_ERROR
      00000000G  00      02  FB 00095      PUSHAB     TMP_RAB
      2A      06  A8  91 0009C  3$:      CALLS      #2, SYSS$PUT
      00      68      0A  13 000A0      CMPL      INP_BUF+6, #42
      20      018C  C8  2D 000A2      BEQL      4$
      68      000A9      CMPCS      INP_REC_SIZ, INP_BUF, #32, #0, INP_BUF
      05      12 000AA      BNEQ      5$
      69      D4 000AC  4$:      CLRL      LINE_TYPE
      0098      31 000AE      BRW      15$
      58      DD 000B1  5$:      PUSHL      R8
      06      DD 000B3      PUSHL      #6
      0000V      CF      02  FB 000B5      CALLS      #2, UPCASE
      0000'      CF      06  29 000BA      CMPL      #6, P.ADQ, INP_BUF
      68      06  12 000C0      BNEQ      6$
      69      02  D0 000C2      MOVL      #2, LINE_TYPE
      0000V      CF      009B  31 000C5      BRW      17$
      01      14  A9  D1 000CD  6$:      CALLS      #0, FREE_FIXED
      35      12 000D1      CMPL      PASS, #1
      31      08  A9  E8 000D3      BNEQ      8$
      BLBS      LINE_SEQ, 8$
      : 3098
```

00	20	68	05	2D	000D7	CMPC5	#5, INP_BUF, #32, #0, INP_BUF	3101
			68		000DC			
		4100	29	13	000DD	BEQL	8\$	
			8F	BB	000DF	PUSHR	#^M<R8,SP>	3106
			05	DD	000E3	PUSHL	#5	
0000V	CF		03	FB	000E5	CALLS	#3, CVT_DTB	
	OC		50	E9	000EA	BLBC	R0, 7\$	
04	A9		6E	D1	000ED	CMPL	TEMP, LINE_NUM	3107
			06	15	000F1	BLEQ	7\$	
04	A9		6E	D0	000F3	MOVL	TEMP, LINE_NUM	3111
		001C8058	0F	11	000F7	BRB	8\$	
E8B8	CF		8F	DD	000F9	PUSHL	#1867864	3114
08	A9		01	FB	000FF	CALLS	#1, SOR_ERROR	
	50	05	01	D0	00104	MOVL	#1, LINE_SEQ	3115
48	8F		A8	9A	00108	MOVZBL	INP_BUF+5, R0	3121
			50	91	0010C	CMPB	R0, #72	3123
	50		05	12	00110	BNEQ	9\$	
			01	D0	00112	MOVL	#1, R0	
49	8F		2F	11	00115	BRB	14\$	
			50	91	00117	CMPB	R0, #73	3124
4F	8F		06	13	0011B	BEQL	10\$	
			50	91	0011D	CMPB	R0, #79	3125
	50		05	12	00121	BNEQ	11\$	
			03	D0	00123	MOVL	#3, R0	
46	8F		1E	11	00126	BRB	14\$	
			50	91	00128	CMPB	R0, #70	3126
	50		05	12	0012C	BNEQ	12\$	
			04	D0	0012E	MOVL	#4, R0	
	01	14	13	11	00131	BRB	14\$	
			A9	D1	00133	CMPL	PASS, #1	3128
		001C8054	0B	12	00137	BNEQ	13\$	
E878	CF		8F	DD	00139	PUSHL	#1867860	
			01	FB	0013F	CALLS	#1, SOR_ERROR	
69			50	D4	00144	CLRL	R0	3121
50			50	D0	00146	MOVL	R0, LINE_TYPE	3134
			69	D0	00149	MOVL	LINE_TYPE, R0	
			03	12	0014C	BNEQ	16\$	
			FE	31	0014E	BRW	1\$	
	0E	28	A9	E9	00151	BLBC	SWITCH_S32, 17\$	3139
	01		50	D1	00155	CMPL	R0, #1	
			09	12	00158	BNEQ	17\$	
19	A8		20	90	0015A	MOVB	#32, INP_BUF+25	3142
1B	A8	58	8F	90	0015E	MOVB	#88, INP_BUF+27	3143
	50		01	D0	00163	MOVL	#1, R0	3146
			04	00166	RET			
			50	D4	00167	CLRL	R0	3147
			04	00169	RET			

; Routine Size: 362 bytes, Routine Base: \$CODE\$ + 1644


```
3093 3148 1 ROUTINE FREE_FIXED: NOVALUE =
3094 3149 2 BEGIN
3095 3150 2 LITERAL
3096 3151 2 NUM = 0;
3097 3152 2 CHR = 1;
3098 3153 2 BIND
3099 3154 2 : Each entry in this table consists of three fields,
3100 3155 2 : The fixed-format position of the field,
3101 3156 2 : The maximum length of the field,
3102 3157 2 : Whether the field is numeric or character.
3103 3158 2
3104 3159 2 H_FIELDS = UPLIT BYTE(
3105 3160 2 1-1, 2, NUM, : Page number
3106 3161 2 3-1, 3, NUM, : Line number
3107 3162 2 6-1, 1, CHR, : Record type
3108 3163 2 7-1, 5, CHR, : Sort to perform
3109 3164 2 13-1, 5, NUM, : Total key size
3110 3165 2 18-1, 1, CHR, : Sort order
3111 3166 2 26-1, 1, CHR, : Collating sequence
3112 3167 2 28-1, 1, CHR, : Keys stripping
3113 3168 2 29-1, 4, NUM, : Maximum record size
3114 3169 2 33-1, 0, CHR, : Comment
3115 3170 2 F_FIELDS = UPLIT BYTE(
3116 3171 2 1-1, 2, NUM,
3117 3172 2 3-1, 3, NUM,
3118 3173 2 6-1, 1, CHR, : Record type
3119 3174 2 7-1, 1, CHR, : Key order
3120 3175 2 8-1, 1, CHR, : Key type
3121 3176 2 9-1, 4, NUM, : Key position
3122 3177 2 13-1, 4, NUM, : Key ending position
3123 3178 2 17-1, 3, CHR, : Force character
3124 3179 2 20-1, 0, CHR); : Comment
3125 3180 2 LITERAL
3126 3181 2 RT_POS = 0, : Position of this field
3127 3182 2 RT_SIZ = 1, : Maximum length of this field
3128 3183 2 RT_TYP = 2; : Type of this field
3129 3184 2 LOCAL
3130 3185 2 BUFFER: VECTOR[80,BYTE],
3131 3186 2 PTR,
3132 3187 2 INP: VECTOR[2],
3133 3188 2 OUT: VECTOR[2],
3134 3189 2 RT: REF VECTOR[3,BYTE];
3135 3190 2
3136 3191 2 : Skip leading spaces
3137 3192 2
3138 3193 2 PTR = CH$FIND NOT CH(.INP REC_SIZ, INP_BUF[0], %C' ');
3139 3194 2 IF CH$FAIL(.PTR) THEN RETURN;
3140 3195 2
3141 3196 2 : Skip digits (page number)
3142 3197 2
3143 3198 2 WHILE (CH$RCHAR A(PTR) - %C'0') LEQU 9 DO 0;
3144 3199 2 IF CH$RCHAR(CH$PLUS(.PTR,-1)) NEQ %C', ' THEN RETURN;
3145 3200 2
3146 3201 2 : Skip digits (line number)
3147 3202 2
3148 3203 2 WHILE (CH$RCHAR A(PTR) - %C'0') LEQU 9 DO 0;
3149 3204 2 IF CH$RCHAR(CH$PLUS(.PTR,-1)) NEQ %C', ' THEN RETURN;
```

```

: 3150      3205 2
: 3151      3206 2
: 3152      3207 2
: 3153      3208 2
: 3154      3209 2
: 3155      3210 2
: 3156      3211 2
: 3157      3212 2
: 3158      3213 2
: 3159      3214 2
: 3160      3215 2
: 3161      3216 2
: 3162      3217 2
: 3163      3218 2
: 3164      3219 2
: 3165      3220 2
: 3166      3221 2
: 3167      3222 2
: 3168      3223 2
: 3169      3224 2
: 3170      3225 2
: 3171      3226 2
: 3172      3227 2
: 3173      3228 2
: 3174      3229 2
: 3175      3230 2
: 3176      3231 2
: 3177      3232 2
: 3178      3233 2
: 3179      3234 2
: 3180      3235 2
: 3181      3236 2
: 3182      3237 2
: 3183      3238 2
: 3184      3239 2
: 3185      3240 4
: 3186      3241 4
: 3187      3242 4
: 3188      3243 4
: 3189      3244 4
: 3190      3245 4
: 3191      3246 4
: 3192      3247 4
: 3193      3248 5
: 3194      3249 4
: 3195      3250 4
: 3196      3251 4
: 3197      3252 4
: 3198      3253 4
: 3199      3254 4
: 3200      3255 4
: 3201      3256 2
: 3202      3257 2
: 3203      3258 2
: 3204      3259 2
: 3205      3260 2
: 3206      3261 2

: It seems to be in free-format. Determine the record type.
SELECTONE CH$RCHAR(.PTR) OF
  SET
  ['F']: RT = F_FIELDS; ! Field specification
  ['H']: RT = H_FIELDS; ! Header specification
  [OTHERWISE]: RETURN; ! Who knows?
TES;

: Make a copy of the original record, and set up descriptors
INP[0] = .INP_REC_SIZ;
INP[1] = BUFFER[0];
OUT[0] = %ALLOCATION(INP_BUF);
OUT[1] = INP_BUF[0];
CH$COPY(.INP_REC_SIZ, INP_BUF[0], %C' ', %ALLOCATION(BUFFER), BUFFER[0]);
CH$FILL(%C' ', %ALLOCATION(INP_BUF), INP_BUF[0]);

: Now loop through the fields in the record, reformatting as we go.
WHILE TRUE DO
  BEGIN
    : Skip blanks
    PTR = CH$FIND_NOT_CH(.INP[0], .INP[1], %C' ');
    IF CH$FAIL(.PTR) THEN EXITLOOP; ! Done if only blanks remain

    : Remove the spaces from the descriptor
    INP[0] = .INP[0] - CH$DIFF(.PTR, .INP[1]);
    INP[1] = .PTR;
    IF CH$RCHAR(.PTR) NEQ %C', '
    THEN
      BEGIN
        : Find the end of this field (a comma)
        PTR = CH$FIND_CH(.INP[0], .INP[1], %C', ');
        IF CH$FAIL(.PTR) THEN PTR = CH$PLUS(.INP[1], .INP[0]);
        INP_REC_SIZ = CH$MOVE( MIN(CH$DIFF(.PTR, .INP[1]), .RT[RT_SIZ]),
          .INP[1], INP_BUF[ .RT[RT_POS] ] +
            (IF .RT[RT_TYP] EQL CHR THEN 0 ELSE
              MAX(0, .RT[RT_SIZ] - CH$DIFF(.PTR, .INP[1])))) - INP_BUF[0];

        : Remove this field from the descriptor
        INP[0] = .INP[0] - CH$DIFF(.PTR, .INP[1]);
        INP[1] = .PTR;
      END;

    : Now scan past the comma
    IF .INP[0] EQL 0 THEN EXITLOOP;
    INP[0] = .INP[0] - 1;
    INP[1] = .INP[1] + 1;
```



```
3207      3262      3      ! Advance to the next field description
3208      3263      3      !
3209      3264      3      ! RT = RT[3];
3210      3265      3      ! IF .RT[RT_SIZ] EQL 0 THEN EXITLOOP;
3211      3266      3      !
3212      3267      3      ! END;
3213      3268      3      !
3214      3269      3      ! Output the reformatted line
3215      3270      3      !
3216      3271      3      ! IF .PASS EQL 1
3217      3272      3      ! THEN
3218      3273      3      ! BEGIN
3219      3274      3      ! PUT_TEXT(
3220      3275      3      !     .LENADR('!'),
3221      3276      3      !     .INP_REC_SIZ, INP_BUF);
3222      3277      3      ! PUT_LINE();
3223      3278      3      ! END;
3224      3279      3      !
3225      3280      1      ! END;
```

```
00 05 0C 01 05 06 01 01 05 00 03 02 00 02 00 0031A P.ADR: .BYTE 0, 2, 0, 2, 3, 0, 5, 1, 1, 6, 5, 1, 12, -
01 00 20 00 04 1C 01 01 1B 01 01 19 01 01 11 00329 5, 0, 17, 1, 1, 25, 1, 1, 27, 1, 1, 28, -
01 01 07 01 01 06 01 01 05 00 03 02 00 02 00 00338 P.ADS: .BYTE 0, 2, 0, 2, 3, 0, 5, 1, 1, 6, 1, 1, 7, 1, -
01 00 13 01 03 10 00 04 0C 00 04 08 00 00 00347 4, 0, 32, 0, 1
21 00353 P.ADT: .ASCII \!\
```

H_FIELDS=
F_FIELDS=

P.ADR
P.ADS

.PSECT \$CODE\$,NOWRT,2

03FC 00000 FREE_FIXED:

```
59 0000' CF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9 : 3148
5E AO AE 9E 00007 MOVAB INP_BUF, R9
52 018C C9 D0 0000B MOVAB -96(TSP), SP
52 20 3B 00010 MOVL INP_REC_SIZ, R2 : 3193
02 12 00014 SKPC #32, R2, INP_BUF
51 D4 00016 BNEQ 1$
57 51 D0 00018 1$: MOVL R1, PTR
01 12 0001B BNEQ 2$ : 3194
04 0001D RET
50 87 9A 0001E 2$: MOVZBL (PTR)+, R0 : 3198
50 30 C2 00021 SUBL2 #48, R0
09 50 D1 00024 CMPL R0, #9
F5 1B 00027 BLEQU 2$
2C FF A7 91 00029 CMPB -1(PTR), #44 : 3199
22 12 0002D BNEQ 3$
50 87 9A 0002F 3$: MOVZBL (PTR)+, R0 : 3203
50 30 C2 00032 SUBL2 #48, R0
09 50 D1 00035 CMPL R0, #9
```

SRTTRN
V04-000

J 5
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 126
(43)

UT I
V04

			2C	FF	F5	1B	00038	BLEQU	3\$		
					A7	91	0003A	CMPB	-1(PTR), #44		3204
			46	8F	11	12	0003E	BNEQ	5\$		
					67	91	00040	CMPB	(PTR), #70		3210
			56		07	12	00044	BNEQ	4\$		
				0000'	CF	9E	00046	MOVAB	F_FIELDS, RT		
			48	8F	0C	11	0004B	BRB	7\$		
					67	91	0004D	CMPB	(PTR), #72		3211
					01	13	00051	BEQL	6\$		
						04	00053	RET			
			56	0000'	CF	9E	00054	MOVAB	H_FIELDS, RT		
			08	AE	52	D0	00059	MOVL	R2, INP		3217
			OC	AE	10	9E	0005D	MOVAB	BUFFER, INP+4		3218
				84	8F	9A	00062	MOVZBL	#132, OUT		3219
			04	AE	69	9E	00066	MOVAB	INP_BUF, OUT+4		3220
0050	8F		69		52	2C	0006A	MOVCS	R2, -INP_BUF, #32, #80, BUFFER		3221
				10	AE		00071				
0084	8F		6E		00	2C	00073	MOVCS	#0, (SP), #32, #132, INP_BUF		3222
					69		0007A				
		OC	BE	08	AE	20	3B	0007B	8\$: SKPC	#32, INP, @INP+4	3231
					02	12	00081	BNEQ	9\$		
					51	D4	00083	CLRL	R1		
			57		51	D0	00085	MOVL	R1, PTR		
					7F	13	00088	BEQL	16\$		3232
		50	OC	AE	57	C3	0008A	SUBL3	PTR, INP+4, R0		3236
			08	AE	50	C0	0008F	ADDL2	R0, INP		
			OC	AE	57	D0	00093	MOVL	PTR, INP+4		3237
				2C	67	91	00097	CMPB	(PTR), #44		3238
					57	13	0009A	BEQL	15\$		
		OC	BE	08	AE	2C	3A	0009C	LOCC	#44, INP, @INP+4	3244
					02	12	000A2	BNEQ	10\$		
					51	D4	000A4	CLRL	R1		
			57		51	D0	000A6	MOVL	R1, PTR		
					06	12	000A9	BNEQ	11\$		3245
			OC	AE	08	AE	C1	000AB	ADDL3	INP, INP+4, PTR	
				57	OC	AE	C3	000B1	SUBL3	INP+4, PTR, R8	3246
				51		58	D0	000B6	MOVL	R8, R1	
51		01	A6	08		00	ED	000B9	CMPZV	#0, #8, 1(RT), R1	
					04	18	000BF	BGEQ	12\$		
			51	01	A6	9A	000C1	MOVZBL	1(RT), R1		
			50		66	9A	000C5	MOVZBL	(RT), R0		3247
			01	02	A6	91	000C8	CMPB	2(RT), #1		3248
					09	13	000CC	BEQL	13\$		
			52	01	A6	9A	000CE	MOVZBL	1(RT), R2		3249
			52		58	C2	000D2	SUBL2	R8, R2		
					02	18	000D5	BGEQ	14\$		
					52	D4	000D7	CLRL	R2		
			50		52	C0	000D9	ADDL2	R2, R0		
		6940	OC	BE	51	28	000DC	MOVCS	R1, @INP+4, INP_BUF[R0]		
				50	69	9E	000E2	MOVAB	INP_BUF, R0		
018C		C9		53	50	C3	000E5	SUBL3	R0, R3, INP_REC_SIZ		
			08	AE	58	C2	000EB	SUBL2	R8		

SRTTRN
V04-000

K 5
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 127
(43)

56		03	C0	000FE	ADDL2	#3, RT	: 3265
	01	A6	95	00101	TSTB	1(RT)	: 3266
		03	13	00104	BEQL	16\$	:
		FF72	31	00106	BRW	8\$	:
01	0000'	CF	D1	00109	16\$:	CMPL	PASS, #1
		16	12	0010E	BNEQ	17\$	: 3271
		59	DD	00110	PUSHL	R9	: 3274
	018C	C9	DD	00112	PUSHL	INP REC_SIZ	: 3276
	0000'	CF	9F	00116	PUSHAB	P.ADT	: 3275
		01	DD	0011A	PUSHL	#1	: 3274
FC78	CF	04	FB	0011C	CALLS	#4, PUT_TEXT	:
FCDB	CF	00	FB	00121	CALLS	#0, PUT_LINE	: 3277
		04	00126	17\$:	RET		: 3280

; Routine Size: 295 bytes, Routine Base: \$CODE\$ + 17AE

UT1
V04

```
3227 3281 1 ROUTINE CVT_BTA (X) =
3228 3282 1
3229 3283 1 ++
3230 3284 1
3231 3285 1 FUNCTIONAL DESCRIPTION:
3232 3286 1
3233 3287 1     Converts positive binary integer to ascii string in the output buffer.
3234 3288 1
3235 3289 1 FORMAL PARAMETERS:
3236 3290 1
3237 3291 1     X           The value to be converted
3238 3292 1
3239 3293 1 IMPLICIT INPUTS:
3240 3294 1
3241 3295 1     NONE
3242 3296 1
3243 3297 1 IMPLICIT OUTPUTS:
3244 3298 1
3245 3299 1     NONE
3246 3300 1
3247 3301 1 ROUTINE VALUE:
3248 3302 1
3249 3303 1     0 or 1
3250 3304 1
3251 3305 1 SIDE EFFECTS:
3252 3306 1
3253 3307 1     NONE
3254 3308 1 --
3255 3309 2 BEGIN
3256 3310 2 LOCAL
3257 3311 2     TMP_BUF:      VECTOR [20,BYTE],
3258 3312 2     Y,
3259 3313 2     IDX,
3260 3314 2     PTR;
3261 3315 2
3262 3316 2     ! Is x valid?
3263 3317 2
3264 3318 2 IF .X LEQ 0
3265 3319 2 THEN
3266 3320 3 BEGIN
3267 3321 3 IF .X EQL 0
3268 3322 3 THEN
3269 3323 4 BEGIN
3270 3324 4 PUT TEXT(LENADR_('0'));
3271 3325 4 RETURN TRUE;
3272 3326 4 END
3273 3327 3 ELSE
3274 3328 3 RETURN FALSE;
3275 3329 2 END;
3276 3330 2
3277 3331 2 ! Initialize index in output string
3278 3332 2
3279 3333 2 PTR = TMP_BUF[%ALLOCATION(TMP_BUF)-1];
3280 3334 2
3281 3335 2 ! Get a local copy of the value
3282 3336 2
3283 3337 2 Y = .X;
```



```
: 3284      3338 2  WHILE .Y GTR 0 DO
: 3285      3339 3  BEGIN
: 3286      3340 4  CH$WCHAR(%C'0'+(.Y MOD 10), .PTR);
: 3287      3341 5  PTR = CH$PLUS(.PTR, -1);
: 3288      3342 6  Y = .Y/10;
: 3289      3343 7  END;
: 3290      3344 8
: 3291      3345 9  PUT_TEXT(
: 3292      3346 0  -CH$DIFF(TMP_BUF[%ALLOCATION(TMP_BUF)-1], .PTR),
: 3293      3347 1  CH$PLUS(.PTR, 1));
: 3294      3348 2
: 3295      3349 3  RETURN TRUE;
: 3296      3350 4
: 3297      3351 5  END;
```

.PSECT \$SPLITS\$,NOWRT,NOEXE,2

30 00354 P.ADU: .ASCII \0\

.PSECT \$CODES\$,NOWRT,2

				0004 00000	CVT_BTA:	.WORD	Save R2	: 3281
				14 C2 00002		SUBL2	#20, SP	
				AC D0 00005		MOVL	X, R2	: 3318
				0A 14 00009		BGTR	1\$	
				39 12 0000B		BNEQ	5\$	: 3321
				CF 9F 0000D		PUSHAB	P.ADU	: 3324
				01 DD 00011		PUSHL	#1	
				28 11 00013		BRB	4\$	
				AE 9E 00015	1\$:	MOVAB	TMP_BUF+19, PTR	: 3333
				52 D5 00019	2\$:	TSTL	Y	: 3338
				15 15 0001B		BLEQ	3\$	
				01 7A 0001D		EMUL	#1, Y, #0, -(SP)	: 3340
				0A 7B 00022		EDIV	#10, (SP)+, R0, R0	
				30 81 00027		ADDB3	#48, R0, (PTR)	
				51 D7 0002B		DECL	PTR	: 3341
				0A C6 0002D		DIVL2	#10, Y	: 3342
				E7 11 00030		BRB	2\$	: 3338
				A1 9F 00032	3\$:	PUSHAB	1(PTR)	: 3347
				AE 9E 00035		MOVAB	TMP_BUF+19, R0	: 3346
				51 C3 00039		SUBL3	PTR, R0, -(SP)	
				02 FB 0003D	4\$:	CALLS	#2, PUT_TEXT	: 3347
				01 D0 00042		MOVL	#1, R0	: 3349
				04 00045		RET		
				50 D4 00046	5\$:	CLRL	R0	: 3351
				04 00048		RET		

; Routine Size: 73 bytes, Routine Base: \$CODE\$ + 18D5

```
3299 3352 1 ROUTINE CVT_DTB(
3300 3353 1     LENGTH,
3301 3354 1     INP_ADR,
3302 3355 1     OUT_ADR:      REF VECTOR[1]) =
3303 3356 1 ++
3304 3357 1
3305 3358 1 FUNCTIONAL DESCRIPTION:
3306 3359 1     Converts decimal ascii string to a binary integer.
3307 3360 1
3308 3361 1 FORMAL PARAMETERS:
3309 3362 1
3310 3363 1     LENGTH Length of input string
3311 3364 1     INP_ADR Address of input string
3312 3365 1     OUT_ADR Address to store the result
3313 3366 1
3314 3367 1 IMPLICIT INPUTS:
3315 3368 1
3316 3369 1     NONE
3317 3370 1
3318 3371 1 IMPLICIT OUTPUTS:
3319 3372 1
3320 3373 1     NONE
3321 3374 1
3322 3375 1 ROUTINE VALUE:
3323 3376 1
3324 3377 1     0 or 1
3325 3378 1
3326 3379 1 SIDE EFFECTS:
3327 3380 1
3328 3381 1     NONE
3329 3382 1
3330 3383 1 --
3331 3384 2 BEGIN
3332 3385 2 LOCAL
3333 3386 2     PTR;
3334 3387 2
3335 3388 2 IF CH$FAIL(CH$FIND_NOT_CH(.LENGTH, .INP_ADR, %C' ')) THEN RETURN FALSE;
3336 3389 2
3337 3390 2 OUT_ADR[0] = 0;
3338 3391 2 PTR = .INP_ADR;
3339 3392 2
3340 3393 2 DECR I FROM .LENGTH-1 TO 0 DO
3341 3394 2     BEGIN
3342 3395 2     LOCAL
3343 3396 2         NUM;
3344 3397 2     NUM = CH$RCHAR_A(PTR);
3345 3398 2     IF .NUM EQL %C' ' THEN NUM = 0 ELSE NUM = .NUM - %C'0';
3346 3399 2     IF .NUM GTRU 9 THEN RETURN FALSE;
3347 3400 2     OUT_ADR[0] = .OUT_ADR[0] * 10 + .NUM;
3348 3401 2     END;
3349 3402 2
3350 3403 2 RETURN TRUE;
3351 3404 1 END;
```


SRTTRN
V04-000

B 6
16-Sep-1984 01:11:52
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTTRN.B32;1

Page 131
(45)

UT I
V04

08	BC		04	AC		000C	00000	CVT_DTB:	.WORD	Save R2,R3	:	3352
						20	3B	00002	SKPC	#32, LENGTH, @INP_ADR	:	3388
						02	12	00008	BNEQ	1\$	:	
						51	D4	0000A	CLRL	R1	:	
						51	D5	0000C	1\$: TSTL	R1	:	
						2E	13	0000E	BEQL	6\$	:	
				OC		BC	D4	00010	CLRL	@OUT_ADR	:	3390
				04		AC	7D	00013	MOVQ	LENGTH, I	:	3393
						1E	11	00017	BRB	5\$	:	
				52		83	9A	00019	2\$: MOVZBL	(PTR)+, NUM	:	3397
						51	D1	0001C	CML	NUM, #32	:	3398
				20		04	12	0001F	BNEQ	3\$	:	
						51	D4	00021	CLRL	NUM	:	
						03	11	00023	BRB	4\$	:	
				51		30	C2	00025	3\$: SUBL2	#48, NUM	:	
				09		51	D1	00028	4\$: CML	NUM, #9	:	3399
						11	1A	0002B	BGTRU	6\$	:	
						0A	C5	0002D	MULL3	#10, @OUT_ADR, R0	:	3400
				OC		51	C1	00032	ADDL3	NUM, R0, @OUT_ADR	:	
				BC		52	F4	00037	5\$: SOBGEQ	I, 2\$	:	3393
				DF		01	D0	0003A	MOVL	#1, R0	:	3403
				50			04	0003D	RET		:	
						50	D4	0003E	6\$: CLRL	R0	:	3404
							04	00040	RET		:	

; Routine Size: 65 bytes, Routine Base: \$CODE\$ + 191E


```
3353 3405 1 ROUTINE UPCASE (LENGTH, INP_ADR): NOVALUE =
3354 3406 1
3355 3407 1 ++
3356 3408 1
3357 3409 1 FUNCTIONAL DESCRIPTION:
3358 3410 1
3359 3411 1 Upcase a string in place.
3360 3412 1
3361 3413 1 FORMAL PARAMETERS:
3362 3414 1
3363 3415 1 LENGTH Length of string
3364 3416 1 INP_ADR Address of string
3365 3417 1
3366 3418 1 IMPLICIT INPUTS:
3367 3419 1
3368 3420 1 NONE
3369 3421 1
3370 3422 1 IMPLICIT OUTPUTS:
3371 3423 1
3372 3424 1 NONE
3373 3425 1
3374 3426 1 ROUTINE VALUE:
3375 3427 1
3376 3428 1 0 or 1
3377 3429 1
3378 3430 1 SIDE EFFECTS:
3379 3431 1
3380 3432 1 NONE
3381 3433 1 --
3382 3434 2 BEGIN
3383 3435 2 LOCAL
3384 3436 2 PTR: REF VECTOR[.BYTE];
3385 3437 2
3386 3438 2 PTR = .INP_ADR;
3387 3439 2 DECR I FROM .LENGTH-1 TO 0 DO
3388 3440 3 BEGIN
3389 3441 3 SELECTONE CH$RCHAR(.PTR) OF
3390 3442 3 SET
3391 3443 3 [X'61' TO X'7A']: CH$WCHAR A(CH$RCHAR(.PTR)-X'20', PTR);
3392 3444 3 [OTHERWISE]: PTR = PTR[1];
3393 3445 3 TES;
3394 3446 2 END;
3395 3447 2
3396 3448 1 END;
```

			0000	00000	UPCASE:	.WORD	Save nothing
	50	08	AC	D0	00002	MOVL	INP_ADR, PTR
	51	04	AC	D0	00006	MOVL	LENGTH, I
			11	11	0000A	BRB	3\$
61	8F		60	91	0000C	1\$: CMPB	(PTR), #97
			09	1F	00010	BLSSU	2\$
7A	8F		60	91	00012	CMPB	(PTR), #122
			03	1A	00016	BGTRU	2\$

```
: 3405
: 3438
: 3439
: 3443
:
```


60208200018
50D60001B 2\$:
51F40001D 3\$:
0400020

SUBB2 #32, (PTR)
INCL PTR
SOBGEQ I, 1\$
RET

: 3444
: 3439
: 3448

; Routine Size: 33 bytes, Routine Base: \$CODE\$ + 195F

; 3397 3449 1
; 3398 3450 1 END
; 3399 3451 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY									
Name	Bytes	Attributes							
\$OWNS	1916	NOVEC, WRT, RD	NOEXE, NOSHR,	LCL, REL,	CON, NOPIC,	ALIGN(2)			
\$PLITS	853	NOVEC, NOWRT, RD	NOEXE, NOSHR,	LCL, REL,	CON, NOPIC,	ALIGN(2)			
\$GLOBALS	72	NOVEC, WRT, RD	NOEXE, NOSHR,	LCL, REL,	CON, NOPIC,	ALIGN(2)			
\$CODE\$	6528	NOVEC, NOWRT, RD	EXE, NOSHR,	LCL, REL,	CON, NOPIC,	ALIGN(2)			

Library Statistics

File	-----			Pages Mapped	Processing Time
	Total	Symbols Loaded	Percent		
_\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	97	0	581	00:01.1
_\$255\$DUA28:[SYSLIB]XPORT.L32;1	590	24	4	252	00:00.6

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS:SRTTRN/OBJ=OBJ\$:SRTTRN MSRC\$:SRTTRN/UPDATE=(ENH\$:SRTTRN)

Size: 6528 code + 2841 data bytes

Run Time: 01:53.7

Elapsed Time: 06:28.1

Lines/CPU Min: 1821

Lexemes/CPU-Min: 26914

Memory Used: 350 pages

Compilation Complete

0367 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500
501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700
701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800
801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900
901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000

0368 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY